INTEGRATING PRESERVATION FUNCTIONS

INTO THE WEB SERVER

by

Joan A. Smith
B.A. 1986 University of the State of New York
M.A. 1988 Hampton University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 2008

Approved by:

_____
Michael L. Nelson (Director)

_____
Kurt Maly

_____
Steven J. Zeil

_____
Mohammed K. Zubair

_____
Simeon Warner

ABSTRACT

INTEGRATING PRESERVATION FUNCTIONS INTO THE WEB SERVER

Joan A. Smith
Old Dominion University, 2008
Director: Dr. Michael L. Nelson

Digital preservation of the World Wide Web poses unique challenges, different from the preservation issues facing professional Digital Libraries. The complete list of a website's resources cannot be cited with confidence, and the HyperText Transfer Protocol (HTTP) provides a bare minimum of metadata with each resource transfer – HTTP is optimized for access *today* rather than *tomorrow*. In short, the Web suffers from a *counting* problem and a *representation* problem. Refreshing the bits, migrating from an obsolete file format to a newer format, and other classic digital preservation problems also affect the Web. As digital collections devise solutions to these problems, the Web will also benefit. But the core World Wide Web problems of Counting and Representation need a targeted solution.

As the host of web content, the web server is uniquely positioned to assist in the preservation of the resources it serves. It recognizes the resources it *has*, and knows what kind of resources they *are*. This dissertation presents research in which preservation functions have been *integrated* into the web server itself to produce *archive-ready* versions of the website's resources. The proposed approach addresses the Counting Problem through the use of Sitemaps, created from a combination of crawling, Sitemap tools, and log analysis. The Representation Problem is addressed by a preservation-preparation module installed on the web server. The module enables each resource to be packaged together with the output from a variety of relevant metadata utilities, creating the aforementioned archive-ready version of the resource. The CRATE Model defines a simple XML structure for the creation and delivery of such resources.

A series of experiments which evaluated CRATE, Sitemaps, and extemporaneous metadata analysis of resources are presented, along with a technical review of the MODOAI web server module which acts as the preservation agent. The feasibility of this approach is demonstrated by a quantitative analysis of its use in a commercial web testing environment.

Dedicated to my mother and to the memory of her mother.

. . . a barren field gives birth to the fertile ground

— Byzantine (Septuagint) Psalter

ACKNOWLEDGMENTS

Considering how many people influenced this research or directly contributed to it through creative discussion, commentary, or insightful critique, it seems unfair to not have them as co-authors or at least co-inspirators, so to speak. If not for Michael Nelson, my eternally patient advisor, this whole project would never have begun. He and Johan Bollen took the time to convince me that pursuing a PhD was a worthwhile endeavor. More importantly, Michael then followed up by securing funding for this research from both the Andrew Mellon Foundation and the Library of Congress. His guidance was always on target; he knew somehow when to apply pressure and when to step back. Then, at the conclusion of my research, he provided the contact with Emory University which has led to a new career path in higher education. I am indebted to him on many levels and for many reasons.

Still, the process is long and courage falters. Stephan Olariu and Hussein Abdel-Wahab provided much-needed encouragement and support at critical points along the way, particularly in overcoming the first hurdle, the Diagnostic (PhD qualifying) Exam. The Candidacy Exam committee members, Kurt Maly, Simeon Warner, Steven Zeil and Mohammed Zubair, provided very helpful feedback and commentary on the initial proposal and I greatly appreciate the time and effort they put into reviewing that document as well as this dissertation.

Throughout these years I've also had the good fortune to have Martin Klein and Frank McCown as project partners, research colleagues and office mates. Working with them has been both productive and rewarding. The pseudo-competition to get papers accepted at conferences added a dimension of fun to what was otherwise just plain hard work. In addition, I have benefited from the software development efforts of Terry Harrison, Aravind Elango, and Ignacio Garcia del Campo, who wrote the initial prototype of MODOAI, from which the author's current modular version was derived; and from the early MODOAI research done by Xiaoming Liu and Nathan MacFarland.

I owe Howard Smith (while at Symantec) and Jim Gray (of Kronos) many thanks for suggesting testing methods and for providing commercial test environments for MODOAI. It isn't often that an academic endeavor such as this can be tested in a commercial environment. The experience was especially helpful in defining criteria for metadata utility compatibility and in designing off-line tests for the software.

When it comes to software engineering, however, one individual stands above the rest. If I have learned anything at all about the art and practice of software engineering, it is thanks to John Owen. As a professional colleague, business partner, and friend, his depth of knowledge and love of the craft have been an inspiration and a guide. The version of MODOAI developed for this dissertation owes its new architecture and implementation style to the debates and discussions we've had. From technical opinions to moral support, his help has been invaluable.

Michael Nelson guided the development of the CRATE concept and its focus on the *counting* and *representation* problems. The complex object implementation, particularly MPEG-21 DIDL, is based on the work of Herbert Van de Sompel and Jeroen Bekaert. Their research significantly helped clarify the concepts and goals of the CRATE model.

There have, of course, been many others whose professionalism and support have made a difference during my time at Old Dominion University. Janet Brunelle gave me an opportunity to teach the Computer Science capstone course, renewing my interest in teaching while simultaneously reminding me just how much work goes into preparing lectures. The department's administrative staff, particularly Phyllis Woods, kept the paperwork straight and the paychecks coming. Ajay Gupta and his systems staff assisted, supported, and restored as only "Sys Admins" can. Among these system administrators, Chris Robinson, Ian Gullet and Joshua Robertson were especially important to the "Counting Problem" experiments. I appreciate their help with both hardware and software.

Ultimately, of course, it all comes down to family, and here I have been blessed with good fortune beyond the normal measure. My parents and siblings have continually expressed both confidence in my abilities as well as a belief in the value of education for its own sake. Last, but most importantly, I have had the unflagging devotion of a wonderful husband. Howard Smith has been with me through it all, with patience and love. There is no way to express how much this has meant to me every day. Thanks to his support and that of many other individuals, this project is finally completed.

None who read it ever wished it longer.
— Samuel Johnson on Milton's Paradise Lost

ACRONYMS

AIP	Archival Information Package (OAIS)
API	Application Programming Interface
CCSDS	Consultative Committee for Space Data Systems
DIDL	Digital Item Declaration Language
DOI	Digital Object Identifier
DIP	Dissemination Information Package (OAIS)
GIF	Graphics Interchange Format
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
JPEG	Joint Photographic Experts Group
LANL	Los Alamos National Laboratory
METS	Metadata Encoding and Transmission Standard
MIME	Multipurpose Internet Mail Extensions
MPEG	Moving Picture Experts Group
NDIIPP	National Digital Information Infrastructure and Preservation Program
NSSDC	National Space Sciences Data Center
OAI	Open Archives Initiative
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
OAIS	Open Archival Information System
PDF	Portable Document Format (Adobe)
PNG	Portable Network Graphics
PREMIS	Preservation Metadata Implementation Strategies
RDF	Resource Description Framework
RSS	Really Simple Syndication
SIP	Submission Information Package (OAIS)
URI	Universal Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	The World Wide Web Consortium
WWW	The World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Digital information lasts forever or five years, whichever comes first.
— Jeff Rothenberg[158]

1 THE CHALLENGE OF DIGITAL PRESERVATION

Significant funds have been devoted to digital preservation research, but there is still no consensus regarding the best preservation strategy. Various governments around the world have organizations whose mission is digital preservation, including the US [131], the UK [129], Holland [97], Australia [130], and Japan [191], among others. Each program has a primary focus, usually the digitization and preservation of official collections such as historical cultural works and/or government records. Some countries are attempting to coordinate their efforts, for example through the International Conference on Preservation of Digital Objects [86], but so far no international standard has been developed and accepted [106].

Preserving quotidian websites is arguably harder than preservation of more formal collections like digital libraries because so many non-professional people are involved in creating and managing their own web content. Such home-grown sites typically lack the metadata and other structures that facilitate archiving commercial digital libraries. A single quotidian website may contain a wide variety of styles and resource types but have no information on who authored the pages and no way to organize the content, such as by topic or by resource type.

Should we care about these ordinary, common-place sites? There are lots of reasons that we should, and not merely because tomorrow's US President probably has pages on today's Facebook site. Much of our social commentary and daily communication has moved from print media to online sites including web pages, blogs, and social networking sites. The one thing all of these websites have in common is that they are hosted on a web server. Since web servers typically host many websites, reaching one server can mean reaching a large number of independent websites. Host servers usually offer a variety of services to their client websites – shopping carts, PHP support, web log analysis – all of which are automated features requiring little to no manpower support by the provider. Many hosting firms also include a simple backup strategy which will automatically restore data lost due to disk failure or other hardware problem. Accessibility failures due to software obsolescence are not usually part of the backup strategy. What if there were an automated preservation option (as opposed to merely a site back-up process), which is simple to

This dissertation follows the style of the *International Journal on Digital Libraries*

install and administer? If we could harness the web server itself as an agent of preservation, we could improve our chances of achieving the long-term preservation of web content, potentially overcoming the threat posed by obsolescence.

1.1 Threats to Preservation

Arguably the most influential document on digital preservation has been the *Reference Model for an Open Archival System* [33]. A comprehensive review of issues and best practices written by the Consultative Committee on Space Data Systems, it identified three key preservation activities:

(1) Data refreshing

(2) Migration from one format to another

(3) Emulation

While future access to the data is certainly a main goal of preservation, there is some disagreement within the preservation community regarding the value of emulation [12], in which a computer environment is used to imitate another, possibly obsolete, environment. Examples include software which emulates old Atari computers and their video games, and Apple's *Rosetta* PowerPC emulator. Regardless of the preservation approach, the problems and issues associated with digital preservation are the same whether the target is a static, small website or a sophisticated, dynamic digital library.

Since digital media is so readily replicated, the need for a preservation strategy may not be obvious. Why not just make a large number of copies, stored in various locations, against the threat of destruction? It turns out that there are many threats to the long-term persistance of digital information. Rosenthal et al. [154] identified over a dozen factors that could limit or prevent recovery (see Table 1). Examples of each can be found in world news archives. As Arms noted in [6], "over a long period of time unlikely events will happen". Merely keeping copies is insufficient, in part because media and their associated hardware (Zip drives, for example) reach obsolescence quickly. In the digital world, preservation encompasses not just *storing* bits and bytes but also ensuring that they continue to be *accessible*.

An accessible archive has little value if it is not also *discoverable*. On the web, content discovery typically means using a search engine like Google or Yahoo. The data that the search engine produces ultimately comes from the web server. A lengthy and repetitive request-response for each URL occurs between the search engine and the web server. The process is often inefficient [29], wasting computational cycles and bandwidth. A bigger problem is that large sections of the web remain "hidden" [150], even though they can be accessed through the server if a visitor knows where to look. Visible content, crawled by search engines, has the advantage of being at least temporarily preserved in the search engine cache, effectively turning the cache into an incidental preservation resource [123]. Discoverability thus aids resource preservation.

TABLE 1: Threats to digital preservation. This list of threat factors for digital preservation comes from [154]. The threats are real. A cursory search of an international news archive will produce an example for each of them.

• Media Failure	• Communication Errors
• Hardware Failure	• Failure of Network Services
• Software Failure	• Software Obsolescence
• Internal Attack	• Operator Error
• External Attack	• Natural Disaster
• Economic Failure	• Organizational Failure
• Media and Hardware Obsolescence	

1.2 Preservation versus Backup

Is there a difference between *preservation* and *backup*? The public recognizes that museums preserve *original* artifacts, and that *replicas* can often be purchased through the museum's store, as in the Rosetta Stone example of Figure 1. Replicas are not usually considered a satisfactory replacement for the original, however. In the digital world, a backup *is* an acceptable replacement [56, 118]. Popular definitions of "backup" can be readily found by doing a Google search on the phrase "define:backup". The following were found in the top 10 Google search results on 16 July 2008:

(A) To create a copy of a disk's contents on another location for safe keeping.[1]

(B) A copy of a file, a set of files, or whole disk for safekeeping in case the original is lost or damaged.[2]

(C) Storing one or more copies of data in case something goes wrong.[3]

These definitions do not have any implied timeline, but describe general safekeeping and do not differentiate preservation from backup. A formal definition of "backup" can be found in [118]:

> "The periodic copying (dumping) of each on-line file to a stable backup storage medium, usually magnetic tape, due to its low cost and high stability."

The U.S. Government's Federal Standard 1037C [180] redirects the reader from "backup" to "backup file", which is then defined as follows:

> A copy of a file made for purposes of later reconstruction of the file, if necessary.
> Note: A backup file may be used for preserving the integrity of the original file and

[1] www.angelfire.com/bc/nursinginformatics/glossary.html
[2] www.geocities.com/mapref/terms/esri_gloss.html
[3] www.crfonline.org/orc/glossary/b.html

may be recorded on any suitable medium. *Synonym* job-recovery control file.

Here, preservation is invoked as contributing to preservation. Colloquially, "backup" and preservation are also closely aligned as the following definitions (taken from the top 10 Google results for "define:preservation") indicate.

(A) [The] activity of protecting something from loss or danger[4]

(B) [The] act of keeping from destruction, decay or any ill[5]

(C) The act or process of applying measures necessary to sustain the existing form, integrity, and materials of an historic property.[6]

There is clearly overlap in the definitions, and no particular distinction made in terms of an expected timeline for the duration of a backup versus a preserved resource. In colloquial usage, the term "backup" seems to convey near-term intent (1 year, for example) while "preservation" implies a very long term intent (100 years, for example). Even so, if an item is preserved for viability 100 years from now, it should be equally viable 1 year from now. Professionals also have some confusion regarding the difference, and whether preservation can be decoupled from use [56]. For the purposes of this research, the author defines "backup" to be a simple copy-and-store procedure, which has no specific strategy to compensate for software or system obsolescence. Preservation is defined as a planned process for the safe-keeping of digital information, including protection from loss, obsolescence and damage, and providing sufficient metadata to facilitate future recovery and usability. Loss and damage protection may be intended by a backup strategy, but the threat of *obsolescence* requires the transition of hardware and file formats often beyond the scope of a backup. The once-ubiquitous 5.25-inch floppy disk has effectively disappeared, and the author's backups of her early software are no longer accessible even though they are merely 20 years old.

1.3 Website Concepts and Definitions

Websites, Web Pages and Web Resources

Websites are a widely-implemented form of digital information which have been in use for over 15 years. What is a website? Readers of this dissertation probably have a general idea of what a website is, but the term has some "fuzzy edges" in popular usage. Consider these definitions, the top five from a Google search of "define: website" on 15 July 2008:

(1) [A] location connected to the Internet that maintains one or more web pages.[7]

(2) A collection of web pages stored on the world-wide-web sharing a common domain name (such as transact.com.au).[8]

[4] http://wordnet.princeton.edu/perl/webwn%3Fs%3Dpreservation
[5] en.wiktionary.org/wiki/preservation
[6] www.loudoun.gov/controls/speerio/resources/RenderContent.aspx
[7] http://www.askoxford.com/concise_oed/website?view=uk
[8] http://www.actewagl.com.au/education/Glossary/default.aspx

FIG. 1: Preservation versus backup. The difference between "preservation" and "backup" is more obvious in the non-digital realm where the concepts of "original"and "copy" are well-understood. Readers know that the images on these pages are themselves copies, and may recognize that the Rosetta Stone [155] in the British Museum (left) is the "original" while the item for sale in a museum catalog [156] on the right is a *replica*, i.e., not the original but which could act as a "backup" for informational purposes even if it is unsatisfactory as a replacement. In digital media, a backup is expected to be an acceptable replacement for the original.

(3) [A] collection of web pages and associated code which forms an integrated presence.[9]

(4) A collection of interlinked documents on a Web server.[10]

(5) [A] collection of Web pages, images, videos or other digital assets that is hosted on one or more web servers, usually accessible via the Internet.[11]

The World Wide Web Consortium (W3C) defines a website as:

> *A collection of interlinked Web pages, including a host page, residing at the same network location. "Interlinked" is understood to mean that any of a Web site's constituent Web pages can be accessed by following a sequence of references beginning at the site's host page; spanning zero, one or more Web pages located at the same site; and ending at the Web page in question* [188].

Most of the popular (Google) definitions incorporate the term "web page" to describe a website. But "web page" is inadequate to describe the wide variety of digital assets accessible via a website. This dissertation uses the term *web resource* rather than web page, and defines a web resource to be the content returned by the server in response to an HTTP request (cf. [188]

[9] www.thedotrepublic.com/terms-conditions.html
[10] ublib.buffalo.edu/libraries/help/glossary.html
[11] http://en.wikipedia.org/wiki/Website

for a similar definition). Resources can therefore be anything from simple HTML documents to multi-media files. Every web resource on a website is a candidate for preservation.

Returning to the issue of website definition, what about "super sites" which aggregate a collection of websites underneath another website? For example, the Computer Science Department at Old Dominion University (ODU-CS) maintains a website at *http://www.cs.odu.edu/* where information about courses, schedules, faculty, and so on are made available to the public. As part of the department's website, every individual with a login to that system has the option of producing his own website as a "tilde subsite" off of the main website. For example, Dr. Michael Nelson's login is "mln" and his website at the university is *http://www.cs.odu.edu/∼mln/*. The department does not maintain the mln website; that is entirely up to Dr. Nelson. The department does manage the supporting infrastructure including server software and hardware. Technically, the mln website is actually a *web subsite*. It could be moved to another server and exist in its own right, and its content is primarily of "type mln", i.e., it "forms an integrated presence" to use the phrase from definition (3). More specifically, and according to the W3C, the mln subsite "is maintained by a different publisher than that of the parent" [188].

Obviously, the definition of website needs to be clarified for purposes of discussion within this dissertation. A website is defined here as a collection of resources having a common domain name, accessible via the Internet. This is slightly more expansive than the W3C definition quoted earlier, in order to take into account web resources that may not be linked from the website's home page (also called "web root"). In practice, this means that if the website is specified as *http://www.cs.odu.edu/* then everything Internet-accessible that begins with "http://www.cs.odu.edu/" is considered to be a part of the website, including any "tilde" portions of the site that are maintained by ODU-CS (there are a few, including an Advising section, *"∼/Advising"*). Similarly, the website specified as *http://www.cs.odu.edu/∼mln/* would only refer to resources that are part of the mln subsite, that is, for which the starting URL is *http://www.cs.odu.edu/∼mln/* and which are maintained by the mln publisher (presumably Dr. Nelson).

URLs and URIs

The term URL has begun to appear in common parlance, having been invented in 1992[12]. Despite its familiarity to non-technical people, the technical definition of URL is not as widely known. URL is an abbreviation of *Uniform Resource Locator*, described by the Internet Engineering Task Force (IETF) in RFC-1738 [21] (updated in RFC-3986 [19]) as being "used to 'locate' resources, by providing an abstract identification of the resource location". Persistent misidentification of the "U" as *Universal* rather than Uniform has resulted in both explanations appearing in publications

[12] http://www.merriam-webster.com/dictionary/URL

and on the web.

URLs have a specific form: **scheme + location details**. The scheme, or *protocol*, for accessing the resource, can be via FTP (File Transfer Protocol), Gopher, electronic mail ("mailto") and others in addition to the commonly-recognized HTTP (HyperText Transfer Protocol). These are all examples of URLs:

(a) ftp://ftp.foo.edu/files/foo/

(b) http://www.foo.edu/index.html

(c) mailto:foo@foo.edu

(d) file://host.foo.edu/note.txt

In (b), the scheme is "http", the location details include the host ("www.foo.edu") and the resource location ("/index.html").

Another abbreviation common in technical literature is "URI," Universal Resource Identifiers [17]. The concept of URI predates URLs. The idea was to define a general namespace organization (URIs) and below that to have specific categories such as URLs, URNs (Uniform Resource Name), and URCs (Uniform Resource Citation). The URC concept was still-born, however, and URNs have not yet seen wide adoption. The net result is that in practice documents sometimes use URL and URI interchangeably. By the definition of URI, this is not correct.

According to the IETF, URLs are a subset of URIs. All URLs are URIs, but not all URIs are URLs. The actual difference is subtle and rarely understood, given the dearth of other URI implementations. The URI was originally defined and described by Tim Berners-Lee in RFC-1630, which he subtitled "A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web". It predates the release of RFC-1780 by six months, during which time additional refinements to URL were introduced. These refinements and the status of RFC-1630 as "informational" have probably contributed to the confusion over URI versus URL. In practice, URI and URL have been used interchangeably for at least 10 years, although there have been suggestions by members of the W3C to unify on a single term[13,14]. Websites may therefore be conventionally described as a collection of either URIs or URLs.

Canonical Resources

Another point to address in digital preservation is the question of the *canonical* version (canonical, derived from theology, means "original" or "source"). Consider this dissertation, for example. Written using a digital typesetting system and authoring tools (LaTeX), it is precisely duplicated on several systems and on a backup DVD. Which of these is the canonical file, or are they all "original"? A small piece of metadata, the timestamp, could possibly be used to declare that one

[13] http://www.w3.org/TR/uri-clarification

[14] http://www.w3.org/Addressing/

particular file is the original and the others are copies, but it is possible to make copies preserving the source's timestamp, confusing the determination of "first" or "original." For websites, the term "canonical URL" can usually be understood as the simplest, most direct path to the resource. For example:

(A) http://foo.edu/fooPage.html

(B) http://foo.edu/?search=fooPage&type=html&date=today

Both URLs are valid, but the *canonical* URL would normally be (A). When attempting to preserve an entire website, canonical URLs are preferred, but they can be difficult to determine [112].

URLs, Web Resources and Resource Representation

The process of retrieving a web resource is straightforward:

$$\text{URL/URI} \longrightarrow \text{Resource} \longrightarrow \text{Representation}$$

A known *location* (URL) is requested via a protocol – HTTP, for example. The request is processed by the server which maps the location to a *resource*, such as a file on the hard drive, or a dynamic script. The server responds to the request by sending a *representation* of the resource back to the client. Why does a distinction exist between resource and representation? The request process allows clients to specify a variety of parameters, including preferred language or file type. The web server's response can therefore vary by the specific format of the request. For instance, *Content-negotiation* in HTTP [83] can determine whether a document, "index.html," is returned in an English-language version or a French-language version. The Apache web server includes a default index page in many languages - English, German, Dutch, and even Czechoslovakian. The page returned to the client can vary by which language the client asks for. An example of such content negotiation is given in Table 2. The server may also return a different version of a resource depending on the type of client making the request. A browser client identifying itself as "Mozilla" could get a slightly different representation than one that has identified itself as "IE" (Internet Explorer). This is in part a concession by webmasters that the representation capabilities of browsers vary.

Up to now, web servers have participated in preservation more by accident than by design. Files that are accessible via the web may be actively replicated on other web servers [22, 79, 40], cached by search engines [123], haphazardly stored by casual users, or intentionally archived as part of a site snapshot [195, 88]. In a previous study we demonstrated that a large percentage of a website's known content could be reconstructed using search engine caches (the "web infrastructure") [122]. Reconstruction only applied to crawled pages, of course. Pages that were not advertised on the site itself or at another site as an external link, were never crawled and therefore were not recoverable from the search engine cache. Our data indicated that an advertised link would be crawled; the key is to advertise every canonical resource to the crawler. By extending

TABLE 2: HTTP content-negotiation examples. Language preference is specified using ISO-639 abbreviations. Media type is specified using MIME abbreviations (discussed in Chapter III, Section 5). For each, the "q" value indicates the order of preference, from 1=highest to 0.1=lowest.

Content Preference	**Syntax**
Get the default document (No language negotiation)	GET /index.html
Get the document in French, if available.	GET /index.html Accept-Language: fr
Get the document in French; if not available in French, then German; if not in German, then English.	GET /index.html Accept-Language: fr, de;q=0.8, en;q=0.7
Get the image (No type negotiation)	GET /fooLunch.gif
Get the image as PNG, if available	GET /fooLunch Accept: image/png
Get the image as JPEG, otherwise get the available format	GET /fooLunch Accept: image/jpeg;q=1, image/*;q=0.5

the interaction between web server and crawler to encompass a more complete view of the site, preservation of web-accessible resources would be improved.

1.4 Other Website Preservation Issues

Website preservation is a much harder problem than it appears at first glance. Some parts of a site may be cloaked from certain clients, and some content may be generated dynamically. These dynamic web resources may be created on-the-fly by the combination of a file system resource (HTML page), data from a database (dynamic content) and content-negotiation by the client. Two sequential website visitors who issued identical request strings could actually receive different content in reply. For example, if the dynamic content selects an image at random from a collection of images, that portion of the page will probably not be the same as it was for the previous response. (The ODU-CS home page currently implements code that does something like this). Add to that the content-negotiation options that can take place during the request/response events, and it is obvious that what appears to be a request for the same resource may not be truly identical.

While a website is sometimes depicted as a directed tree of URIs, in reality it is usually a directed multigraph [23], and it may also have disconnected elements (cf Figure 2).

- Cycles are allowed: A.html → B.html → A.html
- Loops are allowed: A.html → A.html

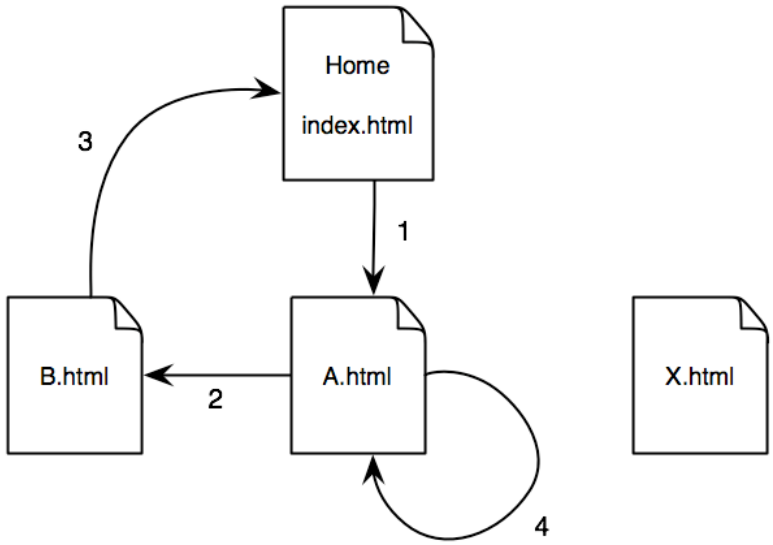


FIG. 2: Website as multigraph. These are sometimes also called pseudographs if they have self-loops [23]. A cycle starts at a page (called a "node" or "vertex" in graph language) and eventually the path (an "edge" in graph language) returns to it, whether via one or several other pages. In this figure, that cycle is represented by the path arrows, 1–2–3 which link the Home page (index.html) to A.html then to B.html, and finally back to Home. A loop points back to itself: many pages also have a link to themselves. This is represented by arrow 4, which starts at A.html and ends there. A disconnected page is not linked to any other page (here, X.html). Compare this graphic with that of Figure 3, which is a file system perspective of a website.

- It may be totally or partially disconnected: X.html

Recall that a URI does not necessarily have a file system equivalent. The loose hierarchy of web pages, including the hypothetical "website tree" pictured in Figure 3, may have only an incidental relationship rather than one based on a file system or internal linking strategy.

Discovery of disconnected resources poses problems for crawlers like search engine robots as well as for users [44]. At some point, the URL must be known or guessed. Both users and crawlers exhibit "guessing" behavior on websites, usually by starting from a known URL and refining the location in an attempt to discover new resources. For example:

Original URL: http://www.foo.edu/dirABC/fooPage.html

Guessed URL: http://www.foo.edu/dirABC/

In many cases, the guessed URL will produce a list of other resources which can then be requested individually. Unlinked URLs may also appear in web logs because the URL creator may attempt to "test" the viability of the resource by accessing it one or more times. In short, enumerating every accessible site resource may not be possible but we may be able to utilize features of the file

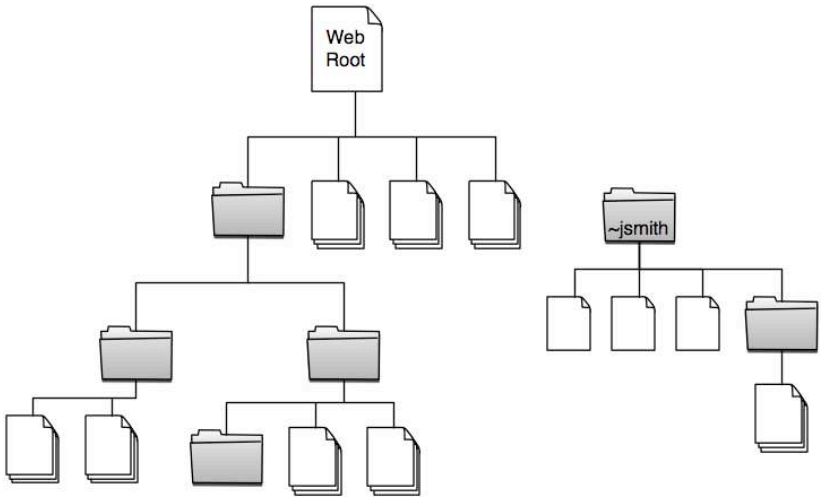


FIG. 3: Website as tree. Websites are often described as "trees", with the website "root" (index.html) linking to resources in other parts of the site, some of which may map to an actual directory or file. In many cases, however, resources exist on the site which are *not* linked to the main part of the website (lower right). In other cases, they may link back-and-forth: websites are more like graphs than trees. This is one reason why a website resource enumeration problem exists.

system, web configuration, and server logs to achieve an acceptable approximation.

2 SCOPE

This dissertation addresses two of the problems that arise in website preservation, the **Counting Problem** (enumerating all of a website's resources) and the **Representation Problem** (describing all of a website's resources). Both are crucial for digital preservation [157], because resources cannot be accessed if we do not know enough about them, and they cannot be preserved if we do not have a copy of them. The web server is uniquely positioned to assist in solving these problems, in large part because it produces the content we seek to preserve. It both *has* the resources and knows *about* them. The following questions represent the scope of this dissertation:

(1) What tools and methods can improve resource enumeration?

(2) Can enumeration tools be integrated with the web server?

(3) Can metadata be automatically and extemporaneously extracted from web resources?

(4) Can metadata utilities be integrated into the web server?

(5) Is it computationally feasible to analyze resources at point of dissemination?

(6) Can the web server perform both resource analysis and resource delivery packaged together in an archival-friendly format?

(7) Can preservation functionality be easily installed on a web server by a typical webmaster?

3 APPROACH

The author designed and ran many experiments during a 3-year period, some of them lasting more than a year. Data was collected continuously throughout the period and the results were monitored for issues such as hardware failure or other errors that tend to occur on live websites. The experiments were focused on obtaining real-life metrics on the activities of web crawlers, such as patterns arising from website change and the time to harvest sites of various type and size. The author also created various experiments examining the impact of metadata utilities on web server performance. The experiments were done in a protected test environment as a proof of concept, followed with longer, more detailed tests on a live, commercial server.

- **The Counting Problem** Several different sets of experiments were conducted to examine website resource exposure, i.e., how much of a site is accessible and, if accessible, how much is actually accessed. Each set involved the creation and installation of at least 4 distinct websites with unique content and a series of scripts to monitor and record results [96, 122, 123, 134, 135, 136, 169, 170, 175].
  - *Search Engine Coverage* The first set of experiments looked at the major search engines, tracking both the breadth and depth of their crawls and their persistence in the face of missing or changed resources. This group of tests ran for over 6 months. A second series of tests were created that looked at site structure and its impact on access. These experiments (4 very large websites of over 20,000 pages each) were conducted over the space of 13 months.
  - *Sitemap Coverage* The author used a snapshot of the Old Dominion University Computer Science Department website to test the usefulness of Sitemaps as a solution for website resource enumeration. The experiment compared the different results obtained from popular tools and the limitations these tools impose.
  - *Tools & Methods for Improving Resource Enumeration* Expanding on the results of the enumeration experiments, additional investigations were performed to examine other potential repositories of website knowledge, i.e., logs and the web infrastructure. An archived copy of Old Dominion University's CS Department website was used as a test bed. The coverage of the website by each source is compared against the original website, and techniques for maximizing coverage are discussed.

- **The Representation Problem** Various utilities are examined which are designed to extract metadata from digital resources. The author's experiments tested a variety of utilities against a mid-sized site having a variety of content commonly found on websites (HTML, PDF, Video, etc.). Detailed performance metrics were obtained for the utilities [168, 171, 172, 173, 174].
  - *Fully Automated* Describes utilities which have a command-line interface and can thus be invoked using scripts. Reviews issues found with attempted implementation of some of these utilities.
  - *Partially Automated* A look at utilities which combine both automatic and manual input.
  - *Web Server-Compatible Utilities* A discussion of the utilities which proved more amenable to server inclusion.
- **Integration of Preservation Functions into the Web Server** The author's central thesis is that the web server can actively participate in website preservation, first by assisting with the process of resource enumeration and second by providing enhanced resource description. It delivers the additional description in a single response together with the resource itself. The reference model for this enhanced representation is called CRATE. A technical implementation of the CRATE reference model was developed and evaluated via a new software module, MODOAI. Although based on a previous proof-of-concept module, the software was completely redesigned and reimplemented by the author over the course of 2 years. The MODOAI module has been installed on several Apache web servers and tested under different load scenarios. MODOAI presents an integrated solution to the two problems of *Counting* and *Representation*. The basis for this model is explained in detail.
  - *Web server modules* Describes the widespread use of web server modules and their installation in the Apache environment. Web server modules are the software foundation for the CRATE implementation approach. (MODOAI).
  - MODOAI A set of experiments was conducted using MODOAI. A mid-sized test website was installed in a commercial, web-testing environment. The results of these experiments are presented and evaluated.
  - *The CRATE Reference Model* Presents metrics of a standard website harvest (just the resources) compared against a full CRATE archiving sequence (resources plus metadata).

4 ORGANIZATION

The dissertation is organized into the following chapters, grouped by major topic. Background material in Chapters 2 – 4 covers basic digital preservation concepts, current practice, and how web services affect preservation. Chapters 5 – 8 present the two preservation problems addressed in this research, the Counting Problem and the Representation Problem. These chapters also describe several experiments conducted as part of this research. Chapters 9 – 11 present the CRATE reference model, an example technical implementation (MODOAI), and an evaluation. Ideas for further research and conclusions are presented in Chapters 12 and 13. Finally, an extensive set of Appendices provides additional technical information about the tools used in the experiments, the CRATE Model schema documents, and other relevant supplementary materials.

The following gives a brief summary of each chapter.

Chapter 2: Current Practice in Digital Preservation Concepts of digital preservation are introduced along with public and private programs aimed at preserving digital content of all kinds. The OAIS model is reviewed, and socio-economic factors that influence preservation are discussed.

Chapter 3: The Current Role of the Web Server in Digital Preservation This chapter looks at the impact the internet has already had on digital preservation, and considers ways in which it can continue to aid preservation, both intentionally and incidentally. Private and public efforts specifically targeting preservation of World Wide Web content are discussed.

Chapter 4: The Current Role of Search Engines in Digital Preservation The concepts of web infrastructure, lazy preservation, and search engines as motivators of web content are examined. A series of experiments which mapped crawler coverage of different websites is reviewed and evaluated.

Chapter 5: Resource Enumeration: The Counting Problem This chapter presents a formal definition of the counting problem and examines why the counting problem exists. The HTTP protocol is reviewed, along with website structure and common methods used for resource discovery.

Chapter 6: Evaluation of Resource Enumeration Methods Describes experiments to test crawling, Sitemaps, and log harvesting as resource enumeration methods. Compares the results of each method, and proposes strategies to achieve maximum website resource listing.

Chapter 7: Resource Description: The Representation Problem Compares the minimal metadata available from an HTTP Response with the breadth and depth of metadata expected in an archival system. Examines some of the many utilities archivists use to generate resource metadata, and their limitations.

Chapter 8: CRATE: A Model for Self-Describing Web Resources The CRATE model is

introduced as a OAIS Submission Information Package. We describe Complex Objects as examples of Archival Information Packages (AIPs). The process of building a CRATE is detailed, followed by a comparison of this model with other complex-object models.

Chapter 9: Evaluation of Metadata Utilities on the Web Server Describes a series of tests implementing utilities in an Apache web server environment. Discusses compatibility of metadata utilities with an operational web server.

Chapter 10: MODOAI: The Integration of Sitemaps and CRATE Describes the design, development, and implementation of an Apache web server module which addresses resource enumertion by using Sitemaps and which implements the CRATE model as a solution for the representation problem.

Chapter 11: Future Work & Conclusions This chapter presents follow-on research areas suggested by the results of the work presented in this dissertation. Summarizes the results and contributions of this research. Discusses advantages of the CRATE approach in web resource harvesting for preservation. Presents the pros and cons of using the web server as an agent of preservation.

Appendices: These provide related materials including CRATE schema documents, examples of the XML text response generated by the Apache MODOAI module, and detailed output from selected metadata utilities.

Vita: The author's curriculum vitae for the 1989 – 2008 time period is included at the end of this dissertation.

CHAPTER II

CURRENT PRACTICE IN DIGITAL PRESERVATION

Data should be "born archival"
— Stewart Brand [28]

1 PRESERVATION: DEFINITIONS AND LIMITATIONS

1.1 Digital Libraries Compared with the World Wide Web

The rapid growth of the World Wide Web ("WWW" or simply "the web") as a phenomenon distinct from the development of professional digital libraries has created a special set of preservation problems, stemming in part from the disorganized nature of the web. Libraries, whether digital or not, have certain characteristics including *unique labelling* of resources, information about each resource, i.e. *metadata*, methods to *locate* and to *retrieve* resources. Some libraries may also function as archives, in which case the resources (or carefully selected resources) have an expected *persistence* over time thanks to *preservation policies* and procedures. Some of these characteristics apply to the web, such as unique labelling: a URL points to one resource, some metadata accompanies the resource, and a scheme (HTTP, FTP, etc.) indicates how to locate the resource. In most other respects the differences are substantial. Even though websites have a specific path to each resource, both the path and the content may change often; multiple paths may lead to identical content; there is little or no metadata about the resources; accessible but unpublished paths are common; and neither the website nor its resources have much certainty of persistence.

Digital libraries have also benefited from research into "best practices" for creating, accessing, maintaining, enhancing and preserving digital information. The seminal work on distributed digital resource collections, is the Kahn-Wilensky Framework (KWF) [93], originally published in 1995 [92]. It described *handle* as the term for a resource's unique identifier, implemented on the web as Uniform Resource Name (URN), and defined the *digital object* as being an abstract data type which is a combination of the *resource* and *key-metadata*, including the URN. Key-metadata may include rights information such as copyright restrictions or limits on distribution. The KWF introduces handle-creation authorities to ensure uniqueness in naming; an access protocol for depositing objects into, and retrieving objects from, a repository; and the aggregation of multiple digital objects into a composite digital object based on some shared characteristic. Figure 4 illustrates these concepts.

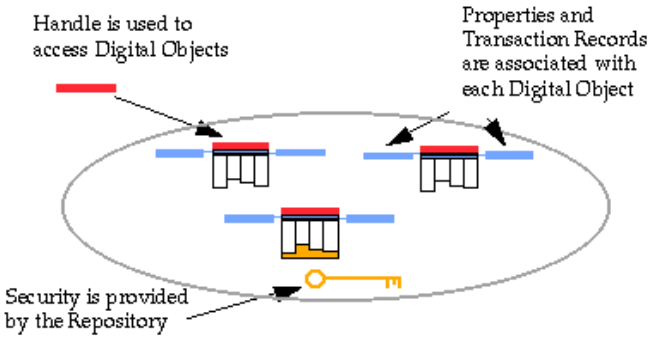


FIG. 4: Conceptualization of the Kahn-Wilensky Framework. A digital library (repository) as conceived in the Kahn-Wilensky Framework where Handle is the access point to the object itself, security is provided by the repository, and metadata is associated with each Digital Object. (Figure from [5]).

In summary, the salient characteristic of a digital library is its managed, metadata-oriented, organized structure which facilitates both current access and long-term preservation. By contrast, even though the web itself has many elements of the KWF such as URNs and URLs, equivalent to KWF handles and the equivalent of handle authorities like the Internet Corporation for Assigned Names and Numbers (ICANN) and Domain Name Servers (DNS), an individual website is often disorganized, lacks metadata, has no inherent rights-management structure, and may literally disappear overnight. Despite superficial similarities between the KWF/digital libraries and the web, they have very different characteristics.

1.2 Metadata Standards

Resources in digital libraries are accompanied by a wealth of auxiliary information known as *metadata*. To quote the U.S. National Information Standards Organization [140],

> *Metadata is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata is often called data about data or information about information.*

In other words, metadata facilitates access. To achieve this facilitation, the metadata must be organized in a way that allows for that information retrieval: in other words, some classification scheme is needed for the metadata. A number of such standards exist.

```
01041cam  2200265 a 450000100200000000300040002000
500170002400800410004101000240008202000250010602000
044001310400018001750500024001930820018002171000032
002352450087002672460036003542500012003902600037000
402300002900439500004200468520022000510650003300730
65000120076 3^###89048230#/AC/r91^DLC^19911106082
810.9^891101s1990####maua###j######000#0#eng##^##$
a###89048230#/AC/r91^##$a0316107514 :$c$12.95^##$a
0316107506 (pbk.) :$c$5.95 ($6.95 Can.)^##$aDLC$cD
LC$dDLC^00$aGV943.25$b.B74 1990^00$a796.334/2$220^
10$aBrenner, Richard J.,$d1941-^10$aMake the team.
$pSoccer :$ba heads up guide to super soccer! /$cR
ichard J. Brenner.^30$aHeads up guide to super soc
cer.^##$a1st ed.^##$aBoston :$bLittle, Brown,$cc19
90.^##$a127 p. :$bill. ;$c19 cm.^##$a"A Sports ill
ustrated for kids book."^##$aInstructions for impr
oving soccer skills. Discusses dribbling, heading,
 playmaking, defense, conditioning, mental attitud
e, how to handle problems with coaches, parents, a
nd other players, and the history of soccer.^#0$aS
occer$vJuvenile literature.^#1$aSoccer.^\
```

FIG. 5: Example of a MARC 21 record. Although the content shown here wraps from one line to another, the record itself has no line ending within it. The numeric elements contain a great deal of bibliographic information in a standardized, but highly condensed form. This example is taken from `http://www.loc.gov/marc/umb/um11to12.html#part11`.

MARC and MARC 21

Machine-Readable Cataloging, or MARC, is one of the earliest automated metadata schemas. [58] It was developed by the Library of Congress to store detailed information about its holdings and has been adopted in whole or in part by many other national libraries including Australia[1], Norway[2], and Korea.[3] A slightly expanded version called MARC 21 which resulted from the merger of similar standards in Canada with those of the USA, is currently used by many English-speaking countries. MARC records can store a high-level of detail, but the basic structure of a record is not easily human-readable. It begins with a 24-character "leader" and is followed by the list of fields and tags (the "directory"), and ending with a 40-character field (the "008" field), all of which is written in one unbroken string. An example of a MARC 21 record is shown in Figure 5.

[1] http://protocat.nla.gov.au/Record/1319301

[2] http://www.nb.no/fag/kompetansesenter/kunnskapsorganisering/dnk

[3] http://www.ndl.go.kr/

TABLE 3: Dublin Core categories. This example uses the Simple (Unqualified) Dublin Core Categories. The hypothetical document is this dissertation. Many fields can be used more than once (Date, e.g.) and some have multiple possible interpretations (Contributor and Creator, e.g.).

Element	**Description/Usage**	**Example (this document)**
Contributor	Editor, translator	Michael Nelson
Coverage	Geographic area; region-encoding	USA
Creator	Author, co-author	Joan A. Smith
Date	File date, creation date, last change date	May 28, 2008
Description	General information	PhD Dissertation
Format	File type, media, manifestation	Adobe PDF
Identifier	Unique ID	∼`jsmit/smith.pdf`
Language	Spoken or written	English (American)
Publisher	Organization, person, originator	Joan A. Smith
Relation	Alternate version(s); e.g. PS, RTF	∼`jsmit/smith.ps`
Rights	Rights held, e.g. copyright and IPR	©Joan A. Smith
Source	Origin of resource	∼`jsmit/smith.tex`
Subject	Classification code, topic, or keyword	Computer science research
Title	Formal name of resource	CRATE: Integrating Preservation Functions Into The Web Server
Type	Content nature, genre	Text

Dublin Core

The Dublin Core Metadata Initiative began as an attempt to create a simple standard set of resource metadata, one that would be easy for anyone to implement [47]. Like many metadata schemes, Dublin Core ("DC") evolved over time and now includes dozens of fields, elements, qualifiers, and options. Table 3 lists the 15 elements referred to as unqualified or "simple" Dublin Core. Despite detailed instructions provided by the DCMI, completion of this set of fields is not simple, and produces different results when several people attempt to describe the same resource, including those with professional training [103, 34]. Defining the metadata for a resource requires domain knowledge, training and experience, even for something as simple as unqualified Dublin Core. Courses in cataloging and classification methods such as Anglo-American Cataloguing Rules, Second Edition (AACR2) are part of Library Science curricula at universities nationwide. Metadata content is still a field for experts.

MODS

When Dublin Core proved inadequate to the task of web resource description, particularly the complex objects found throughout the web, the Metadata Object Description Schema (MODS) was created [72]. Another Library of Congress initiative, it also addresses the need for XML-based metadata exchange. MODS is built on a subset of MARC 21 record metadata, since the full breadth and depth of MARC 21 content was seen as too detailed for a practical XML implementation. It sits between the very simple schema of Dublin Core and the extremely complex schema of MARC.

The Resource Description Framework

The Resource Description Framework (RDF) evolved alongside Dublin Core [125, 151], but exists independently. It is "designed to encourage the reuse and extension of metadata semantics" [125] and as such can contribute to the metadata available for a particular resource. RDF is a W3C recommendation, and part of the semantic web [20], an effort which seeks to map a "relationship web" onto the Internet. RDFa [189] provides guidelines for integrating information into web pages within the conventional HTML tag format. Users can embed information that is both human-usable and machine-scrapable. Like Dublin Core, RDF calls for informed participation rather than amateur input which makes it less likely to be used on pedestrian websites, and which is more typically implemented as part of a professional digital library such as Connexion [43]. Dublin Core, RDF, and other metadata schemes have been incorporated into HTML/XHTML – for example, by using META tags (Dublin Core) or RDFa – but, except for RDF, usage outside of digital libraries is not yet common [66].

2 DIGITAL PRESERVATION MODELS AND IMPLEMENTATIONS

2.1 The OAIS Model

Research into digital preservation predates the Internet by over 20 years. The records from NASA's space program "Voyager" were known to be at risk as early as the 1960s, and the problems were highlighted by the Commission on Physical Sciences, Mathematics, and Applications report in 1995 [42]. The National Space Sciences Data Center formed the Consultative Committee for Space Data Systems (CCSDS), which set up detailed plans to preserve both digital and analog mission data [33]. The CCSDS group's efforts led to the creation of the Open Archival Information System (OAIS), which continues to influence many digital preservation initiatives worldwide.

The OAIS model has been adopted by many digital libraries and preservation projects around

the world. OAIS provides a framework in which all preservation efforts - physical, digital, multimedia - share a common reference. It defines 3 primary roles: producer, consumer, and management. The exact labels may differ by application area - author instead of producer, archivist instead of management, for example - but the concept that items are submitted to an archiving entity for curation and later dissemination is applied to both the physical and the digital worlds.

In the acronym "OAIS" the word "Open" refers to the fact that the framework is developed jointly in an open, public forum, in which any person or group may to participate. There is a detailed Reference Model participants adhere to. The central unit of the Open Archival Information System is the information package itself, which incorporates the object and also other content and supporting information elements such as provenance [33] or reproduction rights. OAIS defines three variations of the information package:

1. **SIP** The Submission Information Package is the item sent by the creator to a OAIS archive for preservation.

2. **AIP** The Archival Information Package is the item packaged by the OAIS archive for preservation, i.e., with descriptive metadata and whatever other elements are necessary for it to endure long-term storage.

3. **DIP** The Dissemination Information Package is the archived item, repackaged and presented to a (future) consumer in a form that makes it usable to that consumer.

In other words, we make, we store, we retrieve. This simple scenario belies the complexity underlying preservation. The canonical OAIS functional model is illustrated in Figure 6, and shows the relationship of the SIP, AIP and DIP elements and the corresponding roles of producer, manager, and consumer. Although the OAIS model encompasses the preservation of both physical and digital objects, the harder problem is generally acknowledged to be in the digital realm [157]. In the OAIS Model, an *information object* results from the combination of a *digital object*, plus the object's *representation information* and any pertinent *knowledge base*. (See Figure 7). To use a real-world example, an HTML page (the digital object) which contains the now-deprecated HTML $< blink >$ tag would need to have representation information (HTML presentation information) and sufficient knowledge base to know that the tagged words should blink when viewed in a browser. Archival Information Packages (AIPs) are expected to contain sufficient (ideally, all) preservation-related data to enable future access, i.e., the representation information and knowledge base information that will be required. The process is so difficult that it has been described as requiring "heroic measures" [105].

A number of tools and procedures have been developed to create submission (SIP), archival (AIP), and dissemination (DIP) packages but their requirements standards differ considerably.

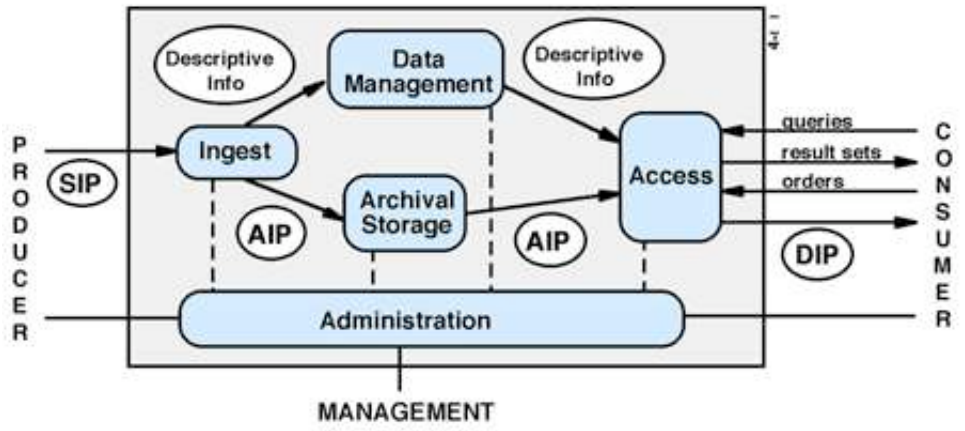


FIG. 6: An operational view of the OAIS functional model. This view of the OAIS Functional Model is taken from the CCSDS Report ([33]). Note the extensive role of management in creating and maintaining the Archival Information Packages (AIP).

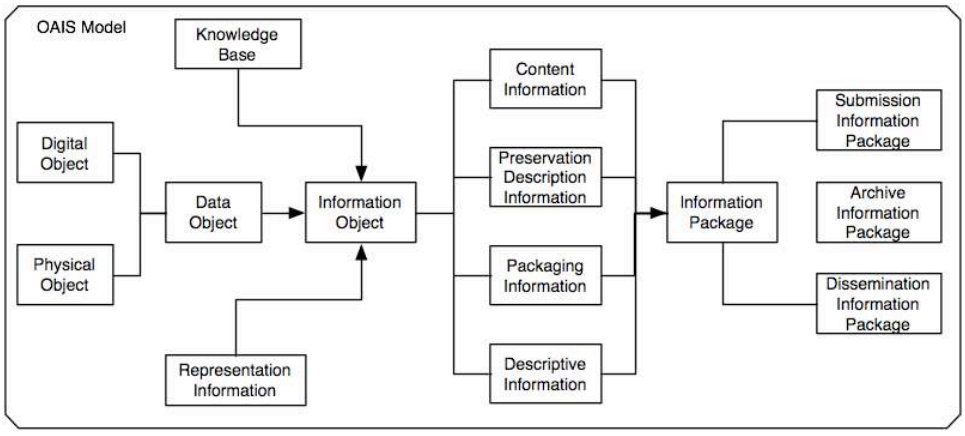


FIG. 7: Objects in the OAIS model. Note the combination of types that contribute to the creation of an Information Object. (Adapted by the author from a figure in [33]).

Victoria Electronic Records System (VERS) Encapsulated Objects are geared specifically to Australian government records [194] and have numerous provenance and signature elements included in them. Trustworthy Digital Objects, described by Gladney, provide mechanisms to ensure long-term verifiability of the object [60]. The Metadata Encoding and Transmission Schema (METS) only requires a file inventory and structural map, with metadata like rights and provenance completely optional [124]. Dublin Core, as mentioned earlier, is geared toward resource discovery. It is not focused on how resources are stored but on having a consistent albeit small set of metadata available [197]. Just cataloging the underlying resource type is difficult, as we will see in Chapter VII, but future accessibility to the resource depends on knowing details like file type, compression schemes, character set, and more. In short, migration and representation are on-going digital preservation problems.

Despite its complexity, the OAIS model has considerable flexibility of interpretation. The OAIS model accounts for the existence of preservation problems, and provides roles and use-cases for addressing them, but it does not prescribe a solution. Individual repositories can implement it as they see fit. As [154] notes, the OAIS model is so flexible that almost any system can claim conformance.

2.2 Complex Objects

The SIP, AIP, and DIP packages of the OAIS model are *complex objects*: they contain not just an item to be preserved, but also include all associated information [132, 137]. One type of complex object that is familiar to the general public is the DVD. The storage model in many cases is the MPEG-4, an encoding format which is a type of complex object implementation. The DVD usually contains not simply a movie but also embedded copyright protection, computer-accessible web links, and occasionally music tracks, among other items. Complex object implementations enable related items to be packaged together in a single digital delivery. The concept is employed in a number of digital preservation systems, such as METS, PREMIS, VERS, and CRATE.

2.3 METS and PREMIS

In the METS model the primary object, a METS "document," contains seven major sections (Figure 8), but only the File and the Structural Map sections are required (see Table 4). The Content element of the File section allows the resource to be included either directly (encoded in Base64) or indirectly (by using a pointer). Indirection lets repositories share information about resources without requiring the resource to be duplicated.

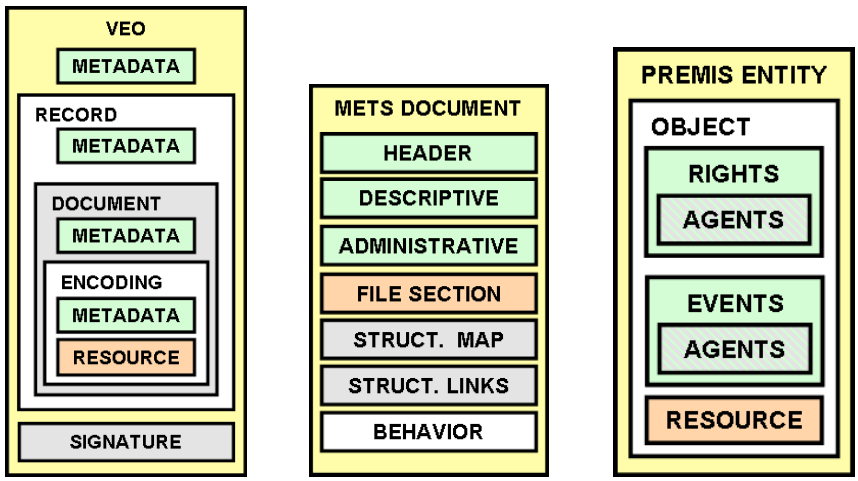
Some metadata schemas have been endorsed by METS. For the Descriptive section, Dublin

TABLE 4: File and structural map sections of METS. The acronym stands for "Metadata Encoding and Transmission Standard."

Section	Element	Examples
File	Location	Source Path Target Path Source URL
	Content	By-Value (Base64) By-Reference (File Ptr/URI)
Structural Map	Div	Sitemap Links In/Links Out

Core and MARC are recommended. For the Administrative section, recommended schemas include the *Schema for Technical Metadata for Text* from New York University, and *Technical Metadata for Digital Still Images* from the National Information Standards Organization (NISO). Like the Victoria Electronic Record System (VERS) ingestion process (see Section 2.4), metadata utilities can be used on each resource to extract data for the Technical portion of the Administrative section. Descriptive metadata is still a problem because neither Dublin Core nor MARC metadata is readily derived.

Repositories customize METS via a *profile* which manages the types of resources it contains. An image collection can have one set of metadata specifications, while audio CD collections have another. Applying METS to a typical website raises complicated issues. Let us assume a hypothetical site containing 3 very different but commonly-found types of resources, HTML, PDF, and JPEG. Using the default profile (from the Library of Congress tutorial website, for example) we would probably need to create three separate METS documents, one for each resource. Alternatively, we could adopt the PREMIS extensions to METS, which is more suited to our sample site. In PREMIS, our website could be mapped to an "Intellectual Entity" with each of the resources comprising an "Object" contained within that entity. Figure 8–(C) gives a conceptual view of the PREMIS Entity. On the other hand, our PDF can be considered a complete Intellectual Entity of its own, and we could therefore archive it as a separate object. In this case, the PDF would have a *relationship* to the HTML referral page. Like many Archival Information Packages, the METS AIP is an XML file where content may be included either By-Value or By-Reference. The structure of the AIP follows the repository's METS profile. In any case, mapping the site's resources to one or more "documents" or to one or more "entities" will depend on the particular implementation at the archiving repository. Two agencies archiving our site could adopt very different strategies and yet adhere to the METS model. The Library of Congress's AIHT Project showed how complicated



(A) (B) (C)

FIG. 8: VERS, METS and PREMIS complex objects. (A) The VEO, a VERS Encapsulated Object; (B) The METS Document Object and (C) the PREMIS Intellectual Entity. Each is a different implementation of the Complex Object type. (All figures from [171]).

ingestion can be when two sources implement a model like METS in different ways [133]. The PREMIS data dictionary addresses this issue by providing more detailed guidelines for metadata fields and content. This is a boon to the knowledgeable archivist, but a daunting set of criteria for the typical webmaster.

2.4 VERS

The Victorian Electronic Records Strategy (VERS) was developed by the provincial government of Victoria Australia to efficiently manage digital versions of official records [192]. VERS metadata objects are designed to ensure authenticity. The VERS workflow (shown in Figure 9) involves the process of converting, encapsulating, and digitally signing each record. A record is stored as an encapsulated object ("VEO") which includes a description of the object format; the VERS version under which it was stored; the digitally signed object, which contains both object content and its metadata; the VEO signature and a locked signature block; and other components designed to certify the content and validity of the record. Details of the VERS Encapsulated Object are provided in a series of implementation guidelines [187]. The process of implementing a pilot VERS system has many steps. These measures may seem extreme for non-official web resources, but forgery of any digital document or website is feasible [165].

The VERS system's focus on evidentiary-quality digital archives requires a great deal more metadata than is currently provided by everyday websites. The most important element, the digital

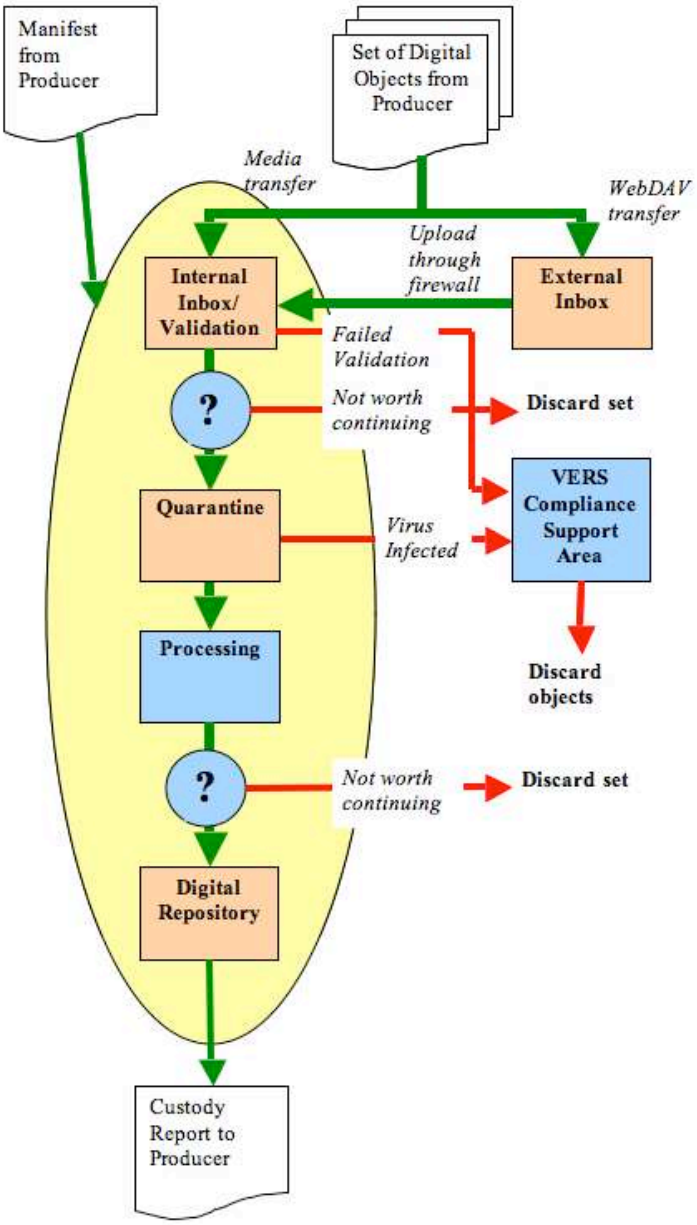


FIG. 9: The VERS workflow. This workflow process was described in [193]. Note the careful processing of the digital object to avoid ingestion of harmful resources. Figure from [193].

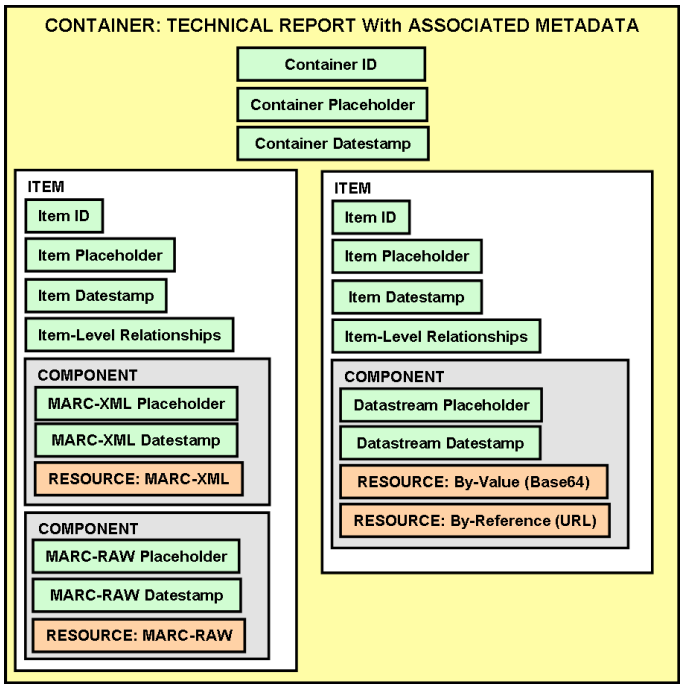


FIG. 10: The LANL MPEG-21 DID complex object. LANL containers can grow wider by having multiple items at the same hierarchical level. They cannot, however, grow deeper since items may not be nested within other items. (Figure from [171]).

signature, poses a problem in that the required PKI infrastructure is not available through many web hosting services; it is further complicated by the need to have a public key on record for this site. In addition, the MD5-Digest directive in the Apache web server defaults to "off," so it must be specifically enabled at the server. Finally, although it is *possible*, digitally-signing HTML documents is not a prevalent practice. JPEG images occasionally have embedded copyright information, but encrypted or digitally signed images are relatively rare. In short, VEOs make good government records (AIPs), but are not practical for quotidian website preservation.

2.5 MPEG-21 DID

LANL has successfully adopted the flexible MPEG-21 DIDL model for use in digital repositories. Figure 10 shows how a Technical Report is stored in the MPEG-21 format at LANL. The main MPEG-21 object, called a "container," can have multiple nested containers, items, and components. "Descriptors" accompany each of these elements to provide information such as origin, date, and element content-type - i.e., metadata about the metadata. Although the original industry specification permitted deep nesting of containers and objects, LANL's implementation only

allows a container to grow in breadth, not depth. This approach simplifies resource access, update, and general management. Like many other XML-based complex-object models, metadata and resources may be included either By-Reference or By-Value. If we have additional information about a resource, it can be included within the container as an additional *item*. For example, more detailed information about file Foo.pdf, including metadata about its embedded images, is produced by the Jhove PDF-HUL module (see Appendix D–1). The Jhove metadata would be contained within one *item-component* in the container, and the Acroread information (Figure 11) would be contained within another *item-component*. A third *item-component* could hold the complete set of response-request fields.

LANL's use of MPEG-21 exhibits a relatively simple ontology consisting of containers which hold items. In turn, items hold 1 or more components (such as MARC metadata). Items may not hold other items, so a LANL container can only grow "wider" as more items are added. Containers are the complex object which aggregates the resource (as one of the items) with its metadata (as one or more additional items in the container). Harvesting a website which has 3 web resources would produce three containers, one per resource. The number of items in each container would vary with the number of metadata source-types. If no utilities were used, only the HTTP metadata item would exist. Otherwise, one item per metadata type would be included in the container. The final element in each container is the resource itself, also enclosed within *item* tags.

3 LOCKSS

Digital preservation solutions often require sophisticated system administrator participation, dedicated archiving personnel, significant funding outlays, or some combination of these. In a similar vein, LOCKSS, "Lots of Copies Keep Stuff Safe," is a solution adopted by several large research institutions and publishers such as Stanford University and the Government Printing Office [117]. Since very long term availability of resources is not guaranteed by publishers, subscribing institutions need a way to ensure long-term access to the information without violating publisher copyrights. An alliance of subscribers which act as a distributed back-up system, LOCKSS provides a collection of cooperative, deliberately slow-moving caches operated by participating libraries and publishers to provide an electronic "inter-library loan" for any participant that loses files. Because it is designed to service the publisher-library relationship, it assumes a level of at least initial out-of-band coordination between the parties involved. Its main technical disadvantage is that the protocol is not resilient to changing storage infrastructures.

The protocol is based on peer-to-peer technology, and is particularly focused on the issue of authenticity, since digital information is easily transformed. Bit-level comparisons of multiple copies are used to ensure file integrity over time. A complicated system of "voting" among the members is used to prevent unauthorized distribution of copies to non-subscribing institutions.

```
<?adobe-xap-filters esc="CR"?>
<x:xmpmeta xmlns:x='adobe:ns:meta/'
  x:xmptk='XMP toolkit 2.9.1-13, framework 1.6'>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:iX='http://ns.adobe.com/iX/1.0/'>
 <rdf:Description rdf:about='uuid:d46586fa-403c-4c1
  <?adobe-xap-filters esc="CR"?>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:iX='http://ns.adobe.com/iX/1.0/'>
 <rdf:Description rdf:about='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:pdf='http://ns.adobe.com/pdf/1.3/'>
  <pdf:Producer>ESP Ghostscript 815.02</pdf:Producer>
 </rdf:Description>
 <rdf:Description rdf:about='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:xap='http://ns.adobe.com/xap/1.0/'>
  <xap:ModifyDate>2007-03-14T10:00:21Z</xap:ModifyDate>
  <xap:CreateDate>2007-03-14T10:00:21Z</xap:CreateDate>
  <xap:CreatorTool>
    dvips(k) 5.95a Copyright 2005 Radical Eye Software</xap:CreatorTool>
 </rdf:Description>
 <rdf:Description rdf:about='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:xapMM='http://ns.adobe.com/xap/1.0/mm/'>
  <xapMM:DocumentID>uuid:ada536f0-811e-487d-b20a-23ebcfe106b7
    </xapMM:DocumentID> </rdf:Description>
 <rdf:Description rdf:about='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <dc:format>application/pdf</dc:format>
  <dc:title> <rdf:Alt> <rdf:li xml:lang='x-default'>jcdl07.dvi</rdf:li>
   </rdf:Alt> </dc:title> </rdf:Description> </rdf:RDF>
</x:xmpmeta>c-9713-43b5a2f3f649'xmlns:pdf='http://ns.adobe.com/pdf/1.3/'>
  <pdf:Producer>ESP Ghostscript 815.02</pdf:Producer>
 </rdf:Description>
 <rdf:Description rdf:about='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:xap='http://ns.adobe.com/xap/1.0/'>
  <xap:ModifyDate>2007-03-14T10:00:21Z</xap:ModifyDate>
  <xap:CreateDate>2007-03-14T10:00:21Z</xap:CreateDate>
  <xap:CreatorTool>
    dvips(k) 5.95a Copyright 2005 Radical Eye Software</xap:CreatorTool>
  </rdf:Description>
 <rdf:Description rdf:about='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:xapMM='http://ns.adobe.com/xap/1.0/mm/'>
  <xapMM:DocumentID>uuid:ada536f0-811e-487d-b20a-23ebcfe106b7
    </xapMM:DocumentID> </rdf:Description>width
  <rdf:Description rdf:about='uuid:d46586fa-403c-4c1c-9713-43b5a2f3f649'
  xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <dc:format>application/pdf</dc:format>
  <dc:title><rdf:Alt> <rdf:li xml:lang='x-default'>draftFoo.dvi</rdf:li>
   </rdf:Alt> </dc:title> </rdf:Description> </rdf:RDF>
</x:xmpmeta>
```

FIG. 11: Acroread metadata. This information is derived from Foo.pdf using Acroread and a technology that Adobe, creators of the Portable Document Format (PDF) call XMP which stands for "Extensible Metadata Platform". An overview of XMP can found at `www.adobe.com/products/xmp/`.

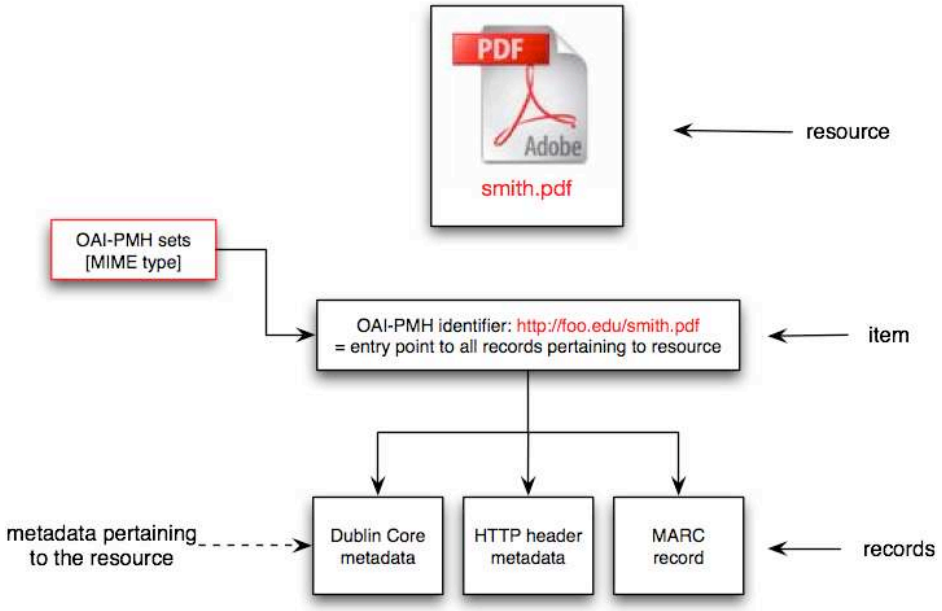


FIG. 12: Basic OAI-PMH data model. This view is adapted from the model presented in [183].

Rights management is a central concern of publishers who have historically depended on the costs and practicality of reproduction as one deterrent to copyright infringement. LOCKSS reassures publishers by tracking which members are authorized subscribers of publications, and allowing restoration of lost materials only when a specific set of criteria has been met.

4 OAI-PMH

The Open Archives Initiative Protocol for Metadata Harvesting, OAI-PMH, is based on a simple data model consisting of *resources, items and records*, as shown in Figure 12. Like many library-driven initiatives, OAI-PMH is focused on resource metadata such as authorship, copyrights, creation and modification dates. Traditionally, the resource itself is not harvested. Instead, queries request *metadata records* such as Dublin Core metadata, using the resource's unique OAI-PMH *identifier* as the point of entry. Each of the metadata records has its own datestamp and identification type; queries can access an item's metadata records by adding qualifiers. If the resource does not change, but one of its metadata records has new information, the resource date will remain the same while the metadata record has the more recent timestamp. Another feature of the OAI-PMH data model is the *set*, enabling selective resource harvesting based not on a resource's location in the site, but on the resource's membership in that set. Finally, responses to queries are

TABLE 5: OAI-PMH verbs. The acronym stands for "Open Archives Initiative Protocol for Metadata Harvesting." It is oriented toward metadata harvesting rather than resource harvesting, but has been adapted to do both. See Figure 13 and the use of Complex Objects.

OAI-PMH Verb	Description
Identify	returns a description of the repository (name, POC, etc.)
ListSets	returns a list of sets in use by the repository
ListMetadataFormats	returns a list of metadata formats used by the repository
ListIdentifiers	returns a list of ids (possibly matching some criteria)
GetRecord	given an id, returns that record
ListRecords	returns a list of records (possibly matching some criteria)

returned in XML, making it easily adaptable to text-based protocols like those used throughout the Internet. For example, OAI-PMH requests and responses are typically communicated over HTTP.

OAI-PMH supports six verbs or "protocol requests" which are listed in Table 5. Three of the verbs are aimed at helping a harvester understand the nature of an OAI-PMH Repository - **Identify, ListMetadataFormats,** and **ListSets**. The ListSets verb can let a harvester know that a site maintains sets, and what those sets are. Resources grouped by MIME[4] types (e.g., image, audio) and subject area (e.g., USHistory, animé) are typical examples of sets that a site might define and support. The other three protocol requests are used for the actual harvesting of XML metadata: **ListRecords** is used to harvest *records* from a repository. **ListIdentifiers** is an abbreviated form of ListRecords, retrieving only *identifiers, datestamps* and *set* information. **GetRecord** is used to retrieve an individual *record* from a repository. Required arguments specify the *identifier* and the *metadata format*.

The URL request string contains all of the elements needed for the server to fulfill the request which is executed via an HTTP GET command. For example, an OAI-PMH repository at baseURL `http://arxiv.org/oai2/` maintains resources in sets called "physics", "cs", "math", and "stat" (among others). To request only the "physics" records, and in particular only those records that have changed on or after 27 September 2006, requires a simple URL:

```
http://arxiv.org/oai2?verb=ListRecords&set=physics
&metadataPrefix=oai_dc&from=2006-09-27
```

The response will contain Dublin Core metadata (*records*) for all items (*identifiers*) in the set "physics" that have changed since September 27th 2006.

An OAI-PMH repository, or data provider, is a network-accessible server that can process the six OAI-PMH protocol requests, and respond to them as specified by the protocol document. A

[4]MIME is discussed in Ch.III, Sect.5

harvester (or service provider) is an application that issues OAI-PMH protocol requests in order to harvest XML formatted metadata. In Representational State Transfer (ReST) terms [53] (over HTTP), this means that cookies or other session-management techniques are not needed. The request string contains all elements needed for processing at the server, and the response string is simple XML over HTTP. Scalability in OAI-PMH is achieved through building hierarchical harvesting networks with *aggregators* – services that are both a harvester and a repository [100]. For example, a site might maintain a collection of PDF files on the subject of "Probability" and also provide metadata (links, summaries) to sites that cover other aspects of statistics.

Since some OAI-PMH requests can result in a very long response, the repository uses a *resumptionToken* to separate the long responses into many shorter responses. A ListRecords response containing 1 million records could be separated into 2000 incomplete lists of 500 records each, which may better suit the load requirements of the server and of the harvester. The fundamental, distinguishing characteristic that separates harvesting with OAI-PMH from regular web crawling is that the repository chooses the size of the Resumption Token, not the harvester. This allows repositories to dynamically throttle the load placed on them by harvesters. The format of the Resumption Token is not specified in the protocol and is left to individual repositories to define. Load-balancing, throttling and different strategies for Resumption Token implementation are discussed in the OAI-PMH Implementation Guidelines [100].

Another powerful feature of the OAI-PMH is that it can support any metadata format defined by means of an XML Schema. The minimum requirement is support for Dublin Core [197], but this metadata set can be automatically derived for a web resource from the HTTP header information. This flexibility has generated considerable interest in liberal interpretations of the data model's elements - *resources, records, and items*. In some cases, it means using OAI-PMH for other than typical bibliographic scenarios [184]. In other cases, the interest is in transmitting the actual *resource* and not just the *metadata*. How does OAI-PMH, which is a metadata-transfer protocol, transmit the resource itself? This is accomplished by encoding the resource itself in Base64 [159, 91]. The resource has been converted to XML-compatible, ASCII-format metadata record which can be included in an OAI-PMH response, as shown in Figure 13.

Despite being a relative newcomer to the list of web-compatible protocols, OAI-PMH is already in use to some extent by Google, MSN, and others [198, 65, 200]. It is an HTTP-based protocol designed to allow incremental harvesting of XML metadata [181], with query responses that are both human-readable and multi-system compatible. The low-barrier nature of the protocol with its six simple verbs and query structure account for the interest by search engines, especially considering the possibility of using a single query to generate a "latest updates" type of response containing records about multiple resources on the server. Such an approach could save a search engine considerable crawling and processing time.

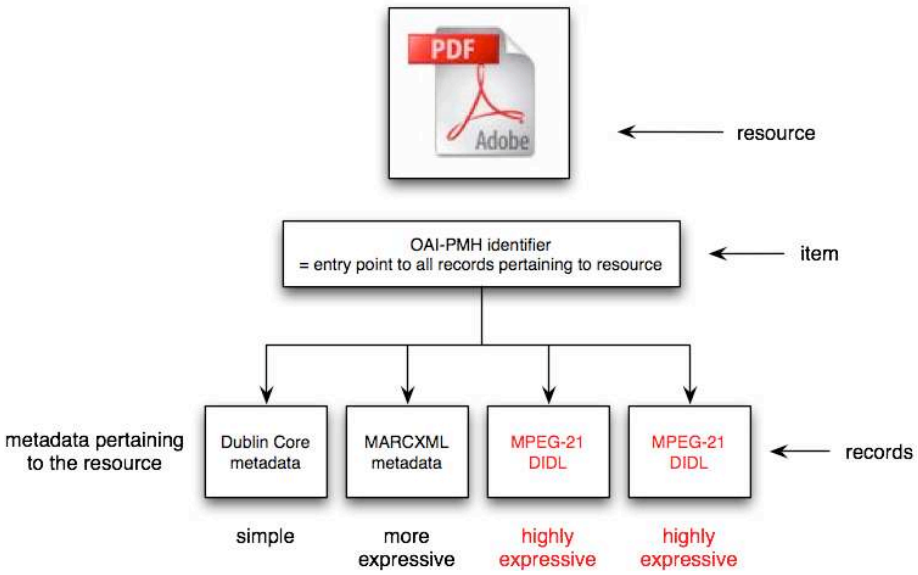


FIG. 13: OAI-PMH with complex objects. The OAI-PMH can include complex objects as metadata, effectively bundling the resource together with its metadata. The MPEG-21 DIDL and METS are complex object types. (Figure adapted from [183]).

5 SOCIO-ECONOMIC FACTORS INFLUENCING PRESERVATION

In his report to the Library of Congress on archiving the web, Peter Lyman cited cultural, economic, and legal problems in addition to technical issues like emulation, migration, and decoding of copyright protections [110]. Many of the problems fall into more than one category. For example, DVDXCOPY software purchased on the web in 2002 is no longer usable because the requisite authorization keys are no longer maintained now that the company is out of business [48]. Law suits caused the company's demise (legal aspect), the software can only be activated by a key found on the original website (technical aspect), and the legal problems arose because commercial DVD producers believed they were losing money to users of DVDXCOPY (economic aspect). For the original DVDXCOPY site, preservation was not an option.

An example of intentional *non-preservation* of information is the cryptography analysis systems developed at Bletchley Park during World War II [166]. Alan Turing's vision of a finite state machine had been successfully built and used to decode ciphers during the war. Rather than risk such advanced technology being stolen, the UK government directed that everything be destroyed – from the plans to the computers and their data [166, pages 279–292]. A similar dilemma was recently raised with the publication of nuclear-bomb fabrication information from an archived copy of an Iraqi website, restored as part of a US government initiative [30]. Another example which

featured prominently in recent headlines is the website of a congressman accused of soliciting (male) Congressional pages [71].

While financial motives may be an obvious factor in the "preservability" of a website, there are other social aspects to consider. Many countries have a ban on certain types of political content, and a wide variety of other topics are considered in bad taste, although the definition of "bad taste" varies by culture. Accessibility to such content via the web would enable external (international) groups to archive sites considered revolutionary by the local government. The decision of *which* sites are formally archived is not easily made, whether by local groups or external third parties. But, supposing that all websites regardless of content are to be preserved, and knowing that the process is linear (at least, to some degree), which sites are at the top of the list, and which are at the bottom? Who should make this determination? When tax dollars are applied to preservation, the public has a vested interest in these decisions.

Governments and organizations necessarily prioritize their efforts because funding for any project is always limited; this constraint applies as much to web archiving as it does to any task. Social, cultural, and political motives will naturally influence web preservation, with "important" collections garnering the lion's share of effort. While the "Top Ten" list of sites may change from one year to another, parochial or pedestrian websites are unlikely to ever be on that list despite the large number of sites created and maintained by ordinary users.

One idea that offers possibilities for improving this situation is to *democratize* archiving. It is the author's belief that putting archiving tools in the hands of the everyday webmaster *could* produce an interest in digital preservation resulting in a variety of today's quotidian sites being accessible in the far future. This dissertation presents concepts and tools that could contribute to this democritization process, and hopefully to more of today's digital information being available in the distant future.

6 SUMMARY

Preservation is a complex problem with no single, perfect solution. Digital information varies too much in format, protocol, and repository requirements to have a comprehensive solution that suits all. Funding, expertise, and infrastructure requirements each impact the feasibility of any given approach. What works well for a highly structured, professionally administered digital library may not be practical for the everyday website and webmaster. Automated approaches *are* reasonable for such websites since they involve little system administration overhead. The complex object concept, which packages the resource and metadata together, could be adapted for use on quotidian sites without involving significant expense or effort.

CHAPTER III

THE CURRENT ROLE OF THE WEB SERVER IN DIGITAL PRESERVATION

Knowledge is of two kinds. We know a subject ourselves,
or we know where we can find information on it.
Samuel Johnson (1709 – 1784)[1]

1 INTRODUCTION

The web server is not currently an active participant in web preservation in the sense of contributing additional, preservation-related information to the archivist about the resources it delivers. For situations where the archiving repository wishes to preserve a website when no direct access to the server (or webmaster) is available, the archivist must use the same techniques that search engines use, i.e., launch a crawler that follows the website's links from a starting point (usually the website root) until all links have been visited, producing a *snapshot* of the website. The crawler gets whatever a normal client would get from the website; there is no additional information imparted by the web server. This is a common scenario, used every day by archiving repositories like the Internet Archive [195]. It was also the basis of the 9/11 preservation experiment, the Archive Ingest and Handling Test, conducted by the Library of Congress [164]. The web server has had a long-standing role in preservation.

2 USENET

In 1997, Hauben and Hauben [77] extolled the value and virtues of Usenet, a collaborative international system of online services accessible via telephone line and modem from virtually any computer in the world. By that point in time Usenet, nicknamed *the Net*, had been already available for about two decades. Even so, it pales in comparison to the Web we have today. Once the Net became *the Web* i.e, the World Wide Web, many of the servers that had provided its Bulletin Board Services (*BBS*) were no longer available. With them went many notable Net items, like the announcement of the World Wide Web from Tim Berners-Lee [64] and the first posting mentioning "MS-DOS." [63]

Remarkably, much of this was recovered thanks to a concerted effort by Google to restore these postings and to make them accessible to the Web community [62]. What makes this remarkable is that these files were not recovered from a preservation archive but from the backup copies made by *everyday users* when Usenet was in its prime. In some cases, the data came from Exabyte

[1] From Boswell's Life of Johnson `http://www.gutenberg.org/etext/1564`

(A) Languages on the Rosetta Stone (B) Rosetta Stone metadata as Dublin Core

FIG. 14: Languages on the Rosetta Stone. Three languages appear on the Rosetta Stone. While all were in common use when the stone was made, by the time of archeological discovery only ancient Greek was still understandable. Just as ancient Greek succeeded as an international language of scholarship, ASCII has succeeded as the basis of computerized character encoding. The metadata of the Stone was a key to the representation and understanding of other ancient languages. (A) is from [178]; (B) is from [50]

and DAT tape media of users; in other cases, the data was restored from backup CDs in personal collections; and a large set came from the archives of a group purchased by Google, Deja News, which had played a big role as a provider of information for Usenet clients [102].

There were many factors contributing to the success of the Usenet project. One was that the timeline between the end of Usenet and the attempt to recover was relatively short, so the hardware needed to extract information from things like DAT tapes could still be found, although with difficulty. But perhaps the biggest reason that it was possible to restore so much of the former Usenet is that the data was primarily in plain ASCII text rather than in specialized character sets or in a proprietary binary format. ASCII, the American Standard Code for Information Interchange, was developed in the 1960's to represent standard English-language characters and selected control codes for machine-processing instructions [153]. ASCII or an extended variant of ASCII (i.e., UTF-8) is still used worldwide [201]. Just as Greek was the deciphering key provided on the Rosetta Stone (Figure 14), ASCII was they key to deciphering the bits on the various media.

The restoration of Usenet appears to fit the description of "heroic measures" of preservation that Levy mentioned [105]. This example stands as a kind of exception to the rule that preservation is done by preservationists, but it also serves as a hint of future opportunities if preservation is democratized. In addition, it suggests that *simple* formats and clear encoding could increase the

likelihood of long-term preservation of digital information. Finally, it is an example of the role plain luck can play in preservation success.

3 THE INTERNET ARCHIVE

In the United States, the task of website preservation has been largely assumed by the Internet Archive (IA). Although it receives some funding from the Library of Congress, Internet Archive is a private, philanthropic endeavor funded primarily by Brewster Kahle which collects snapshots of websites [195], not just once but several times over the course of years. The IA hosts a site called the *Wayback Machine*[2] where historical snapshots of sites can be viewed. As of this writing, the Wayback Machine has historical views of the Old Dominion University Computer Science Department website dating back to 1997 (Figure 15). The snapshot for a particular date can range from only a few pages to nearly complete, depending on what the crawler was able to access at the time the site was visited by IA, and by the capabilities and archiving resources available to IA.

As a typical web crawler, IA has many of the same limitations that other search engines have, i.e., it depends on the web server for information [31, 37]. Getting a complete listing of *possible* and *accessible* URLs at a site is no easy task [29, 78]. For web preservation this means that refreshing is a problem: unfound resources are unrefreshed resources; uncrawled resources are unpreserved resources. Even those that *are* refreshed lack sufficient forensic metadata for preservation. The IA is also biased to small, popular formats and preserves relatively few audio and video files.[3]

There are often long delays between Internet Archive crawls and the posting of a site snapshot, and sites that have frequent changes are unlikely to have all possible snapshots captured. Sites that have been crawled and archived at the IA may nonetheless prove inaccessible, or be missing key elements and images that were part of the original site. The URLs taken from the Wayback Machine in Table 3 demonstrate some of the problems users can experience when attempting to access snapshots. These example URLs reflect historical information that is only a few years' old, but which is already incomplete. Despite the Internet Archive mission of web preservation, it is a primarily philanthropic endeavor funded by Brewster Kahle with some additional funding from organizations like the Library of Congress. As such, its income is much lower than Google's billions of dollars in capitalization. It is therefore unrealistic to expect the IA to succeed in archiving more than a small fraction of web content.

Many websites have been replicated by users around the web, sometimes intentionally (the Comprehensive TeX Archive Network – CTAN – mirrors, for instance); sometimes illegally (the Russia-based clone of O'Reilly resources at the former site `http://www.orelly.com`, for

[2] http://www.archive.org/web/web.php
[3] http://www.archive.org/about/faqs.php

FIG. 15: Archived ODU-CS webs on the Wayback Machine. The Internet Archive's Wayback Machine (`http://web.archive.org/web/*/http://www.cs.odu.edu`) has taken numerous snapshots of the Computer Science Department (ODU-CS) website over the years. Not all those shown can actually be accessed, and links within snapshots may also be broken.

TABLE 6: Problematic snapshots on the Wayback Machine

Path index error: `http://web.archive.org/web/19971010201632/http://www.cs.odu.edu/`
Missing content: `http://web.archive.org/web/20030419163818/http://www.cs.odu.edu/`
Not in archive: `http://web.archive.org/web/19970606105039/http://www.cs.odu.edu/`

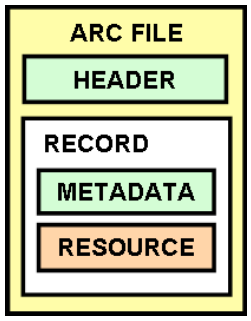
example) [40]. Search engines have been known to cache large portions of a site, but our experiments showed that usually such a cache is purely temporary: if the site disappears, the cached copy on the search engine usually follows soon after [123]. Site mirroring is not a solution for long-term web preservation, in part because it plays a backup role rather than a historical archive role [22]. Once site components are changed, the mirror typically reflects the change (else it is not a *mirror*), so the evolution of a file through its various incarnations would usually also be lost. Site mirroring probably has limited utility as a preservation tool for the long haul. Even if a mirrored site was preserved in its bit-wise splendor, figuring out how to interpret it at some distant future point would be a challenge. Preservation demands keeping enough related information to enable sensible future access.

The Internet Archive stores crawled sites in a file format called *ARC* [107], shown in Figure 17-(A), with a command-line example shown in Figure 17-(B). Except for the protocol headers, web crawling using HTTP, FTP, and NNTP typically generates little or no explicit descriptive metadata. A website merely needs to be crawled by the Alexa robot for the ARC file to be created, or it could use IA's Heritrix [127] to self crawl, creating an archival-quality snapshot of the site.

Such an approach does not provide much in the way of future forensic information, so the Internet Archive also offers an expanded preservation-oriented crawling service, *Archive-It*. [4] The service is on a fee-based subscription, and allows the subscribing site to provide Dublin Core metadata, multiple "seed" URLs, varying schedules for each seed, and other archiving details. For our sample site, we would need to manually introduce the Dublin Core information for each resource, via the Archive-It catalog form. Even though this is an improvement forensically over plain HTTP metadata, expressing technical information in these fields is awkward, at best. Consider the Jhove analysis of our JPEG resource, shown in part in Figure 16. What parts of the analysis should be entered into Dublin Core fields? What kind of consequences arise from discrepancies in the output from other utilities if we do choose to include some or all of the information?

```
<?xml version="1.0" encoding="UTF-8"?>
<jhove xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://hul.harvard.edu/ois/xml/ns/jhove"
      release="1.1" date="2006-06-05">
   <date>2007-04-19T12:20:23-04:00</date>
   <repInfo uri="/var/www/Barfoo.jpeg">
      <reportingModule release="1.2" date="2005-08-22">JPEG-hul
         </reportingModule>
      <lastModified>2007-02-03T18:22:23-05:00</lastModified>
      <size>25474</size>
      <format>JPEG</format>
      <version>1.01</version>
      <status>Well-Formed and valid</status>
      <sigMatch><module>JPEG-hul</module></sigMatch>
      <mimeType>image/jpeg</mimeType>
      <profile>JFIF</profile>
      <property>
         <name>NisoImageMetadata</name>
         <values arity="Scalar" type="NISOImageMetadata">
         <value>
            <mix:mix xmlns:mix="http://www.loc.gov/mix/"xmlns:
                  xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.loc.gov/mix/mix.xsd">
            <mix:BasicImageParameters>
               <mix:Format> <mix:MIMEType>image/jpeg</mix:MIMEType>
               <mix:ByteOrder>big-endian</mix:ByteOrder>
               <mix:Compression>
                  <mix:CompressionScheme>6
                  </mix:CompressionScheme>
               </mix:Compression>
               <mix:PhotometricInterpretation>
                  <mix:ColorSpace>6</mix:ColorSpace>
               </mix:PhotometricInterpretation>
               </mix:Format>
            </mix:BasicImageParameters>
            <mix:ImageWidth>459</mix:ImageWidth>
            <mix:ImageLength>253</mix:ImageLength>
            <mix:Energetics>
            <mix:BitsPerSample>8,8,8</mix:BitsPerSample>
         </value>
      </property>
   </repInfo>
</jhove>
```

FIG. 16: Jhove metadata. This is part of the metadata derived from Barfoo.jpeg using Jhove's JPEG-HUL module. See Appendix D for other Jhove sample output.



(A)

```
filedesc://IA-001102.arc 0 19960923142103
text/plain 76 1 0 Alexa Internet
URL IP-address Archive-date
Content-type Archive-length



http://www.dryswamp.edu:80/index.html
127.10.100.2 19961104142103 text/html 202
HTTP/1.0 200 Document follows
Date: Mon, 04 Nov 1996 14:21:06 GMT
Server: NCSA/1.4.1
Content-type: text/html
Last-modified:
Sat,10 Aug 1996 22:33:11 GMT
Content-length: 30
<HTML>
Hello World!!!
</HTML>
```

(B)

FIG. 17: ARC model. Figure (A) shows the conceptual view of an ARC object and its components. Text in (B) shows ARC file example data (sample content from `http://www.archive.org/web/researcher/ArcFileFormat.php`).

ARC files are plain ASCII text, and any characters outside that range must be “escaped”. Whether we use the expanded, Dublin Core-based version or the original, the archived file conforms to the ARC format: file header with version information followed by the URL record which begins with a list of metadata fields included with this particular record, and ends with the actual content returned from the HTTP method (e.g., GET). ARC files are compressed at both the URL-record level and at the file level, for improved storage. Although not written in XML, an ARC file is mostly human-readable, once uncompressed, as shown in Figure 17.

The International Internet Preservation Consortium (IIPC) has developed an extended revision of the ARC format called “WARC” (for “Web ARChive”) which lets harvesting organizations aggregate large amounts of web resources into specific collections with locally-assigned metadata such as “subject” or unique record ID. The proposed WARC format has numerous sections to clearly delineate “records” in the file. A record, in WARC terms, can be the “response,” the “request”, file structure (“warcinfo”), or other descriptive information. Like other managed collection models, WARC expects the repository to provide any metadata outside of the HTTP request-response event information. This can be a challenge for the average web master.

4 OTHER WEB ARCHIVING EFFORTS

Website preservation is the mission of organizations like Japan's National Diet Library [191] and the National Library of Australia Digital Services Project. [130] The U.S. has funded the National Digital Information Infrastructure Preservation Program (NDIIPP) which, in turn, funds preservation research and programs [131]. The mission of the Henry A. Murray archive at Harvard University [75] is to "preserve in perpetuity all types of data of interest to the research community", much of which is now web based. Both Holland [97] and the United Kingdom [129] have national programs that are attempting to preserve their national digital heritage, and include at least some portion of the Web in the program's scope. In addition, the European Archive is a relatively recent endeavor whose goal is to "[lay] down the foundation of a global Web archive based in Europe." [49] The primary focus at this point is European heritage, including non-web-based materials. Despite these efforts, as noted in Chapter II, there is no effective *global* web-preservation strategy in place, nor is there likely to be, given the wide variation in goals and the expense involved in such an undertaking. Like the Usenet experience, the casual, quotidian website is more likely to be preserved by accident than by design.

5 THE WEB SERVER AND FORMAT MIGRATION

The technologies of digital file formats evolve just as they do in hardware design. PostScript files, once the only way to achieve a truly "typeset" look for a printed digital document, have since been almost completely replaced by Adobe's Portable Document Format (PDF). Amazon's e-book reader, the Kindle [1], can enable text and images to be resized, linked, bookmarked and annotated, all while maintaining a book-like experience for the reader. The underlying file format is ".mobi", which was specifically designed to support mobile devices that have small form factors (i.e., the viewing screens are much less than letterpaper in size). Amazon bought the rights to the .mobi format and added some proprietary extensions, including Digital Rights Management (DRM), producing a native Kindle format, file type ".azw". Will this become the next "PDF" – that is, will documents transition to an interactive format used by e-book readers? This format is relatively new, but it is currently supported by popular magazines (Forbes, Newsweek, and Time for example), a selection of national and international newspapers (Le Monde, Shanghai Daily, and the New York Times, among others), and many publishers. As of this writing, over 100,000 book titles are available in the format, with more being added daily. For these publications, the Kindle content is nearly identical with the print version, whereas the on-line versions are generally quite different. Despite the relative newness of the format, it is gaining rapid acceptance. The publishing divisions of several major universities (Yale, Princeton, and the University of California among others [89]) have made their course text materials available on the Kindle. Whether the growth of

mobile book sales will influence Web formats or other aspects of web content delivery is yet to be seen.

Format convenience could play a role in .mobi adoption for more than just Kindle use. As an experiment, the author used the Amazon conversion service to transcribe an early draft of this dissertation, accomplished by sending the PDF file to a special Amazon email address where it is reformatted and then sent wirelessly to the owner's Kindle. For this experiment, the LaTeX code was compiled without the *hyperref* package which automatically activates links in the document. Amazon states that the Kindle conversion service is still experimental at this time, and in fact the conversion did not produce a fully "Kindle-ized" document: Clicking on an entry in the table of contents brings the reader to that point in the document, but bibliographic references and footnotes are not linked, for example, whereas they are linked in the professionally-produced files purchased from Amazon. For the second experiment, the author recompiled the document with the hyperlinks active in the PDF. This feature includes the ability to link from footnote number to footnote content, page reference number to page, and so on. When converted to the .mobi format and sent to the Kindle, all of the linking remained intact; the document had the look-and-feel of documents purchased from Amazon for the Kindle. This conversion/migration is an intriguing example of a nearly-interactive migration process for digital documents arising directly from web services.

Other factors, from commercial to philanthropic, may play a role in digital format transitions. For example, Microsoft has recently introduced the *XPS* format as an alternative to PDF. An abbreviation of XML Paper Specification, XPS "describes electronic paper in a way that can be read by hardware, read by software, and read by people"[4]. Like other file format transitions, its success will depend at least in part on consumer adoption rates.

On the web, HTML exists alongside XHTML and dynamic content generation tools like Javascript [51, 111]; content presentation techniques for web clients continue to evolve. How does a browser recognize what to do with the content it receives? Web servers and browsers use the Multipurpose Internet Mail Extensions, MIME, to identify the *kind* of file found in the response. Originally defined to facilitate email message exchange [24], MIME was also adopted for use in HTTP [18] and so has been a part of the web since the beginning.

MIME types are expressed in HTTP using a simple two-part formula of *Content-Type*/*Content-Subtype*. The nine Content-Types currently defined by IANA are Application, Audio, Example, Image, Message, Model, Multi-part, Text, and Video. For each of these, one or more Content-Subtypes are defined. For example, the Image Content-Type has dozens of Content-Subtypes defined such as "cgm" (Computer Graphics Metafile), "jpeg" (Joint Photographic Experts Group), "png" (Portable Network Graphic), and "vnd.microsoft.icon" (Microsoft icon image file). The

[4] http://www.microsoft.com/whdc/xps/default.mspx

latter Subtype uses the "vnd" prefix to indicate that it is a vendor-specific Content-Subtype, although this designation does not prevent the Subtype from being used for non-Microsoft icon images. The combination of Content-Type and Content-Subtype, including the ability to add such vendor-specific labels, enables MIME to keep up with the evolution of file formats.

One feature of HTTP is that it can use MIME to specify the range of formats that a client will accept from the server. The "Accept" request-header field lets the client say that it prefers to receive an image as "png", but will take other formats, as in this example:

```
Accept:  image/png;q=0.9, image/*;q=0.2
```

The asterisk in "image/*" is how the server indicates that it will accept any MIME image format. See Table 2 on page 9 for other examples.

Browsers use MIME identification to determine how the content will be handled for the client. File types can be expressed by the dot extension of the file name (.pdf, .html, .txt, etc.) and by a special byte sequence at the start of the file. Accidentally naming a PDF file "file.doc" instead of "file.pdf" does not change the file type, which can possibly be determined by examination of the byte signature of the file itself. The Internet Assigned Numbers Authority, IANA, is responsible for managing the registration of MIME types [85], but web servers and browsers are individually responsible for recognizing the various MIME types and for defining handling – content transformation and presentation or "representation information" as it is called in OAIS terminology (cf. Chapter II, Section 2.1). Because MIME recognition and handling is locally configured, a particular MIME type may not be recognized by the web server, the client, or both.

Consider the example in Figure 18. Browsers may have plugin modules which can understand a wide variety of file types. Interpretation of content is limited only by the availability of a plugin, and sufficient interest in the file type is likely to push for the development of such a plugin. The author's current version (2.2.3) of the Apache web server has over 700 MIME types, which is more than 10 times the number that were defined in the Apache version installed in the year 2000. The web server's role is merely to send the response, and having a large database of recognized MIME types does not add significant overhead to the server host. On the other hand, browsers are interpreting responses for display on the client system. The more MIME types a client handles, the larger the software module. It is then no surprise to find that a browser does not support a MIME type which is no longer in widespread use. MIME itself does not provide insight into technical details of the file type. Only utilities which examine the file contents can provide that depth of information. Long-term preservation requires more than the web server currently delivers.

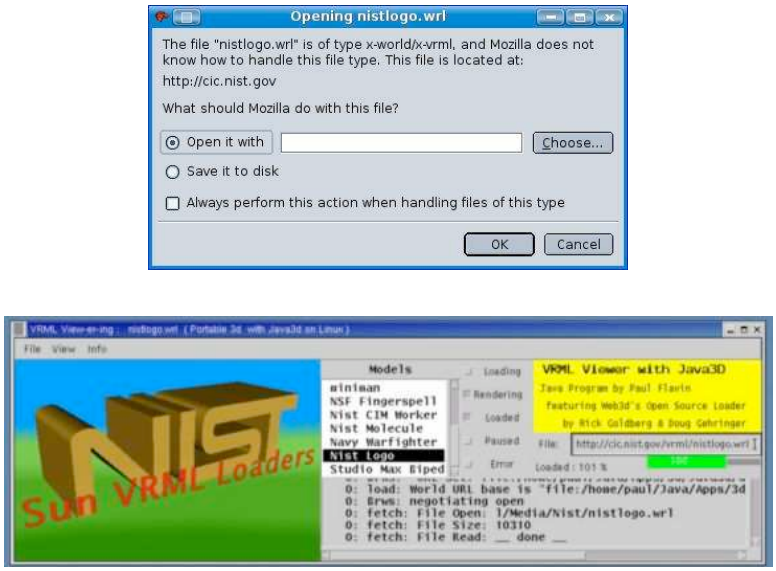


FIG. 18: Unhandled MIME type. On the top is the error message generated by the author's web browser when attempting to get the VRML file, *nistlogo.wrl* (a 3-D image). The web server knows the MIME type, but the browser has no plugin to handle it. It defaults to a decision box asking what to do with the resource. In OAIS terms, there is insufficient *Representation Information* and *Knowledge Base* for this Digital Object to produce an *Information Object*. On the bottom is another web server [57] where the file has been converted into something currently understandable by the browser. The received MIME type in this case is image/jpeg, which the browser displays without problem.

6 SUMMARY

The benefits of "preservation by popular demand" have already been seen in the Usenet project undertaken by Google, where the combined heroic efforts of professionals and hobbyists have produced a nearly-complete restoration of the once ubiquitous service. The Internet Archive's efforts have proved important to many people, and have helped restore websites around the globe [123]. More importantly, they have brought the issue of preservation of web content into the popular domain. Still, a single organization, whether national or international cannot match the sheer volume of web content awaiting an archivist. Grass-roots preservation efforts and personally archived collections of popular sites may be what provides the future with information on this generation's digital heritage. MIME typing of files may be able to identify the general resource type but it does not provide a migration path nor give insight into details that determine the proper representation

of the file. In OAIS terms, it may not contain sufficient Representation Information about the Digital Object to produce an Information Object, particularly if the appropriate Knowledge Base is also missing. For that information, other utilities need to be applied to the resources. The current web server response is inadequate for long-term preservation.

CHAPTER IV

THE CURRENT ROLE OF SEARCH ENGINES IN DIGITAL PRESERVATION

So many men, so many questions. (Quot Homines Tot Sententiae)
Terence (185 BC – 159 BC)[1]

1 THE SEARCH ENGINE AS AGENT OF DISCOVERY

Both Digital Libraries and websites have a vested interest in encouraging search engine robots to thoroughly crawl their sites, even if the content requires a subscription or access fee. Many repositories such as those of the IEEE and The New York Times allow selected crawlers full site access so that the information will be indexed and listed in search results, potentially bringing new subscribers or item-purchasers to the sites [29, 161, 121]. The information is thus made visible even to non-subscribers, who typically get directed to an abstract or summary page, from which point they have an option to purchase the item or subscribe in full. Such users may not take advantage of facilities like local university libraries but instead rely completely upon Google, Yahoo, and MSN (the "Big Three") to find this information, whether it is free or fee-based [144]. For competitive reasons, search engines want to find content, and they provide guidelines to webmasters for improving "findability" of site resources. Google, for example, makes specific recommendations with regard to site organization, number of links per page, and the use of a Sitemap [68], but it will nonetheless crawl sites that do not follow their guidelines.

1.1 Observations of Web Crawler Behavior

How do crawlers approach an everyday, not-so-famous site? Are they equally thorough in their crawls? Does the design of the site – deep or wide – affect robot behavior? Are there strategies that increase crawler penetration? These are important questions because crawler penetration equates to accessibility and therefore to likelihood of *replication* (as in the Usenet example), and from there to eventual preservation. Research into crawler behavior has usually focused on either building smarter crawlers [78, 37, 40], or on improving site performance and accessibility to crawlers [29, 150]. In contrast, the author designed a series of experiments to ask: *Given certain website designs, how do crawlers perform?* The first set of experiments monitored the impact of site content removal on crawler behavior. The second set of experiments looked at the role site depth, breadth, and link structure played in crawler penetration and rate of coverage. Each of these is reviewed in detail in the sections that follow.

[1] As quoted in Adagia by Erasmus.

1.2 Crawler Behavior on Websites with Disappearing Content

The "Decaying Directories" experiments [123, 134, 170] began with a 30-directory wide website containing both HTML and PDF files as well as a collection of images (PNG, JPEG, and GIF). Four of these experimental websites were created, each hosted at a separate website. The four sites are labelled FMC, JAS, MLN, and OBR. In part the goal was to simply monitor the request patterns by the crawlers as content on the site disappeared gradually over a 90-day period. Another goal was to evaluate the persistence of website content in the cache of each of the three search engines. This requirement meant that the site should be less than 1000 resources overall, because some search engines restrict queries against the cache to less than 1000 per day.

Each website was organized into a series of *update bins* (directories) which contained a number of HTML pages referencing the same three inline images (GIF, JPG and PNG) and a number of PDF files. An index.html file (with a single inline image) in the root of the website pointed to each of the bins. An index.html file in each bin pointed to the HTML pages and PDF files so a web crawler could easily find all the resources. All these files were static and did not change throughout the 90 day period except (a) selected files were *erased* on a specific schedule and (b) the index.html files in each bin were modified by having links to the deleted web pages removed.

The number of resources in each website was determined by the number of update bins B, the last day that resources were deleted from the collection T (the *terminal day*), and the bin I which contained 3 images per HTML page. Update bins were numbered from 1 to B, and resources within each bin b were numbered from 1 to $\lfloor T/b \rfloor$. Resources were deleted from the web server according to the bin number. Every n days one HTML page would be deleted (and associated images for pages in bin I) and one PDF file from bin n. For example, resources in bin 1 were deleted daily, resources in bin 2 were deleted every other day, etc. We also removed the links to the deleted HTML and PDF files from bin n's index.html file.

At any given day d during the experiment (where $d = 0$ is the starting day and $d \leq T$), the total number of resources in the website is defined as:

$$Total_c(d) = 2 + \sum_{i=1}^{B} Total_b(i,d) \tag{1}$$

The total number of HTML, PDF and image files in bin b on any day d is defined as:

$$Total_b(b,d) = HTML(b,d) + PDF(b,d) + IMG(b,d) \tag{2}$$

The total number of resources in each update bin decreases with the bin's periodicity as shown in Figure 19. The number of HTML, PDF and image files in each bin b on any day d is defined as:

$$HTML(b,d) = \lfloor T/b \rfloor - \lfloor d/b \rfloor + 1 \tag{3}$$

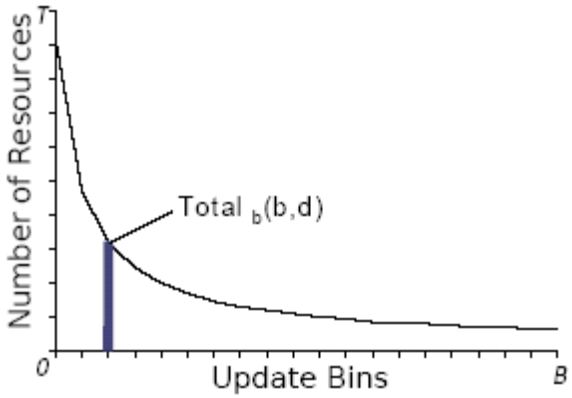


FIG. 19: Number of resources in the test website

$$PDF(b,d) = \lfloor T/b \rfloor - \lfloor d/b \rfloor \tag{4}$$

$$IMG(b,d) = \begin{cases} 3(HTML(b,d)-1) & \text{if } b = I \\ 0 & \text{if } HTML(b,d) = 1 \\ 3 & \text{otherwise} \end{cases} \tag{5}$$

In each website, 30 update bins were created ($B = 30$) that completely decayed by day 90 ($T = 90$), with bin 2 ($I = 2$) containing the supplemental images. So the total number of files in each collection on day 0 was $Total(0) = 954$. Each of the websites was limited to less than 1000 resources in order to control the number of daily queries to Search Engine (SE) caches, based on restrictions imposed by those SEs.

Website Design and Implementation

To ensure that each site would have unique information content, the author designed a set of HTML and PDF files using a randomized English dictionary. Although individual words might repeat between pages, word phrases of 5 or more words remained unique. The subdirectories linked directly to their own content, which consisted of up to 226 files (a combination of HTML, PDF, and images). Figure 20 illustrates the website structure; Table 7 lists the distribution of content by type and byte size. In all, 4 unique websites were created and published on the same day and within a few minutes of each other using a suite of site authoring and launching tools created by the author.

The experiment was planned to last 120 days, with the content on the each site designed to last only 90 days. The Time-To-Live (TTL_{ws}) for each resource in the website is determined by its bin

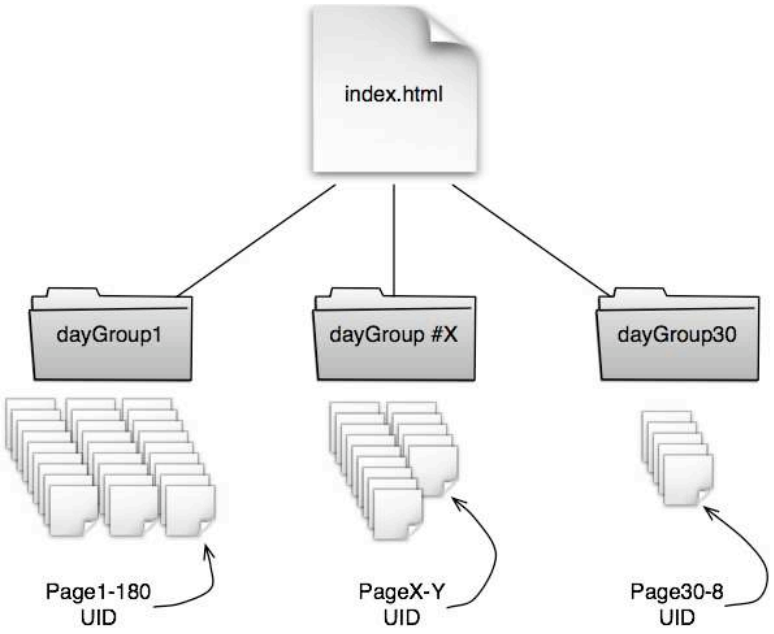


FIG. 20: Decaying directories structure. The number of resources in each subdirectory varied. In subdirectory 1, a resource was deleted every day; in subdirectory 15, a resource was deleted every 15 days, etc. By the end of the 90-day experiment, all of the directories are empty. In other words, they gradually decay to just an index page, with no links; even the subdirectories are removed.

TABLE 7: Website content of the Decaying Directories experiment. The average sizes refer to experiment start, i.e., before any decay has occurred. Obviously, the average size of a site decreases with each day's resource removal. Resources are actually *erased*, not simply unlinked. Otherwise, "remembering" a link would produce an HTTP 200 response (OK) rather than the desired 404 response (not found).

Qty	Description	Avg Size
31	index pages	48 KB
350	random-content HTML pages	735 KB
350	random-content PDF files	41650 KB
74	PNG images	4662 KB
75	JPG images	1063 KB
74	GIF images	518 KB
954	URIs per site	48,676 KB

number b, page number p, and the website terminal day T:

$$TTL_{ws} = b(\lfloor T/b \rfloor - p + 1) \tag{6}$$

After the 90-day point, only the index pages in each of the directories would remain and all other content would have been removed. Crawler behavior was monitored for a full 30 days beyond the removal of the last subdirectory. The author wrote a script to automatically remove one or more resources from the site every day. The total content of the directories thus "decays" downhill over the course of the experiment, as shown in Figure 21. The removal process included erasing all links to those pages from other parts of the site, ensuring that all link references were up to date. Each of the sites was installed within an existing website (technically making the experimental web a "subsite" of the host website [188]). Website "discovery" was not a direct factor, because each of the sites had been visited by Google, MSN, and Yahoo at least once before the experiment began. However, the daily deletion of one or more resources from each of the websites mean that it was virtually impossible for a crawler to harvest the entire website.

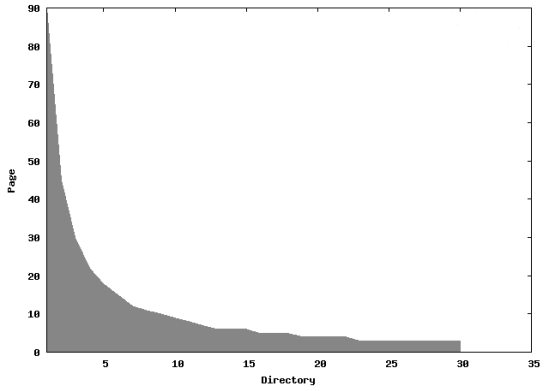
Data Collection

Logs for all of the sites were harvested for a six-month period, starting 2 months before initialization with the new content and ending one month after the last experimental resource was removed. From this monitoring, the author was able to confirm that Google, Yahoo, and MSN had visited the host websites at least once before the experiment began. It was not therefore necessary to inform the crawlers of the existence of these sites, since each would find the new links upon their next crawl of the host website main page.

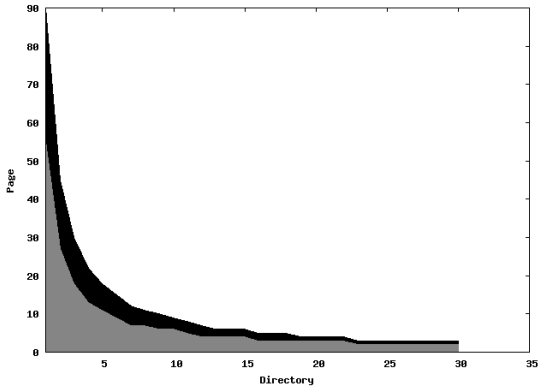
The author wrote a series of utilities to harvest the logs specifically for crawls of the experimental resources. The data from each crawl was mapped to an X,Y data point, where X represented the subdirectory number (from 1 to 30, with 0 being experiment root) and Y represented the resource number. The visits from each of the crawlers could be mapped onto a graph of site resources, visually communicating the process of crawling the site. Similarly, the presence of any site resource in a search engine cache could also be mapped onto the same graph.

Profiling the search engines was complicated by the limited data collected by the site web logs, which were not directly owned by the author. Only the OBR site tracked user-agent information, which is how Google, MSN, and Yahoo identify themselves to the web server. The lowest-common-denominator of metadata available in the logs was remote host IP address, timestamp of the request, the request itself (which contains the HTTP method, URL, and HTTP version used), the status response returned to the requestor, and the number of bytes sent in the response. The difference can clearly be seen in the two examples from the JAS and OBR web logs shown

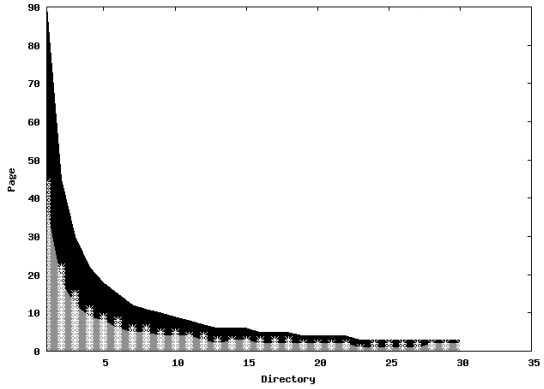
FIG. 21: Gradual resource removal. Resources are removed gradually over a 90-day period. The foreground (gray) represents "live" resources, while the darker background (black) indicates those web resources that have been removed. Note the downhill decaying pattern as the experiment day number increases. The crawl data is mapped on top of the disappearing resource picture as "snowflakes" (white star-pattern points in the foreground). An animation of the crawl showing the gradually disappearing site with crawler activity as the snowflakes can be seen in [170].



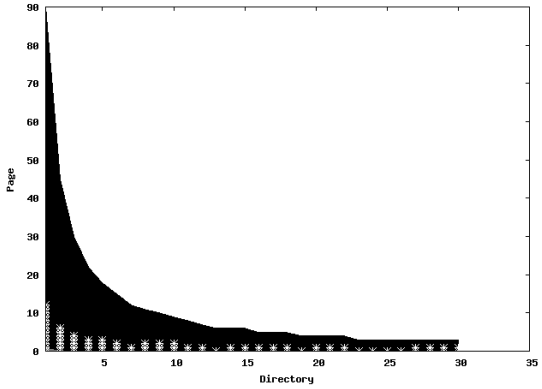
Website Day 01



Website Day 36



Website Day 54



Website Day 90

TABLE 8: Log data from two of the Decaying Directories sites. The data clearly show how the amount of information available from a web server log can vary. Dash in the column indicates the field is tracked but empty for this record; N/A means the field is not tracked by the server. For an explanation of the fields, see the Apache Guide ([2]).

Log Field	OBR Site Log Entry	JAS Site Log Entry
host (IP)	66.249.66.69	207.46.98.59
ident	-	-
Authuser	-	-
date	[26/Jun/2005:17:44:42 -0400]	[05/Jul/2005:00:14:06 -0400]
request	"GET /dgrp12/index.html HTTP/1.1"	"GET / jsmit/dgrp16/index.html HTTP/1.0"
status	200	200
bytes	1237	815
server name	www.owenbrau.com	N/A
uagent	"Mozilla/5.0 (compatible;Googlebot/2.1; +http://www.google.com/bot.html)"	N/A

in Table 8. For this experiment, the important *uagent* field is only tracked at the OBR site. As a result, extra processing had to be done to the logs to lookup the host IP address and determine which visitor had crawled each resource. Why the difference in data logged by the servers? Web logging takes both processing time and disk space. For busy sites, reducing the fields tracked in the web logs can save them time and money.

One characteristic common to experiments performed on live websites is that service problems will occur. On occasion, logging at one or more of the sites would fail, or a firewall parameter would be reset leaving the site temporarily inaccessible. These issues particularly affect the OBR site, to the point that data from the site was so scant as to be useless. Another problem arises when visitors are not identified (the uagent field is not tracked), because determining authoritatively who owns the visitor IP address is not always possible. In some cases, the "log-resolve" and "whois" databases (which are used to determine the identity of the visitor) do not match precisely. For example, the IP address 207.68.61.254 is attributed to Verizon in log-resolve, but whois says that Microsoft owns the IP. This is not unusual, since resale of IPs to other business units is well-documented. The author opted to use the DNS entry of record as the final arbiter.

Other identification problems arose which were harder to resolve. Yahoo acquired Inktomi in December 2002, well before this experiment began. Presumably, Yahoo kept the Inktomi-named robots, since no Yahoo-named robot crawled any of the test collections, but the visited pages showed up on Yahoo's site in direct relation to the pages crawled by robots named Inktomi. The author therefore treated Inktomi robots as Yahoo robots. Google, Inktomi/Yahoo, and MSN

comprise the bulk of log records (nearly 80%) for non-spam search engines.[2] Traffic was minimal from other popular engines like "Picsearch" and "Internet Archive", with only 157 total requests from Internet Archive and 315 from Picsearch, i.e., less than 1% of all search-engine requests. Table 9 summarizes the number of crawled pages, by search engine.

TABLE 9: Crawler statistics from the Decaying Directories experiment. Because of persistent accessibility problems with the hosting service of the OBR site, there was insufficient data for analysis.

	Total Requests by Site			
Crawler	FMC	JAS	MLN	OBR
Google	2813	3384	3654	162
MSN	768	780	808	0
Inktomi	991	1735	1569	49
Picsearch	29	152	134	0

Crawler Characteristics

Search engines employ a large number of systems to make their crawls through a site. Some, like Google, associate a unique remote host name with each robot (IP address). Others (notably MSN) may use numerous IPs but still resolve to only one remote host name. For purposes of this experiment, it was sufficient to aggregate the requests by search engine rather than IP. What happens at the remote host site is unknown, of course, but the point of these experiments was to watch a search engine's pattern as a whole, rather than the pattern of each individual robot. The patterns for all three of the primary crawlers were similar: Request one or two index pages on the first day, then traverse a majority of the site on the following day(s). This was termed a "toe dip" since it was similar to a swimmer testing the water before plunging in. This behavior can be seen for each of the sites in animated graphs in [170]. A summary view of the coverage for the MLN site is shown in Figure 22. The relative request rate for each of the major search engines appears nearly constant, compared with the graph of ODU's own robot. It requested more resources, more often, than any of the major search engine robots. An hourly overview of the request patterns for Google, MSN, Yahoo and ODU can be seen in Figure 23.

[2] A spam search engine is one which attempts to harvest email addresses and data from websites for use in developing web spam or email spam.

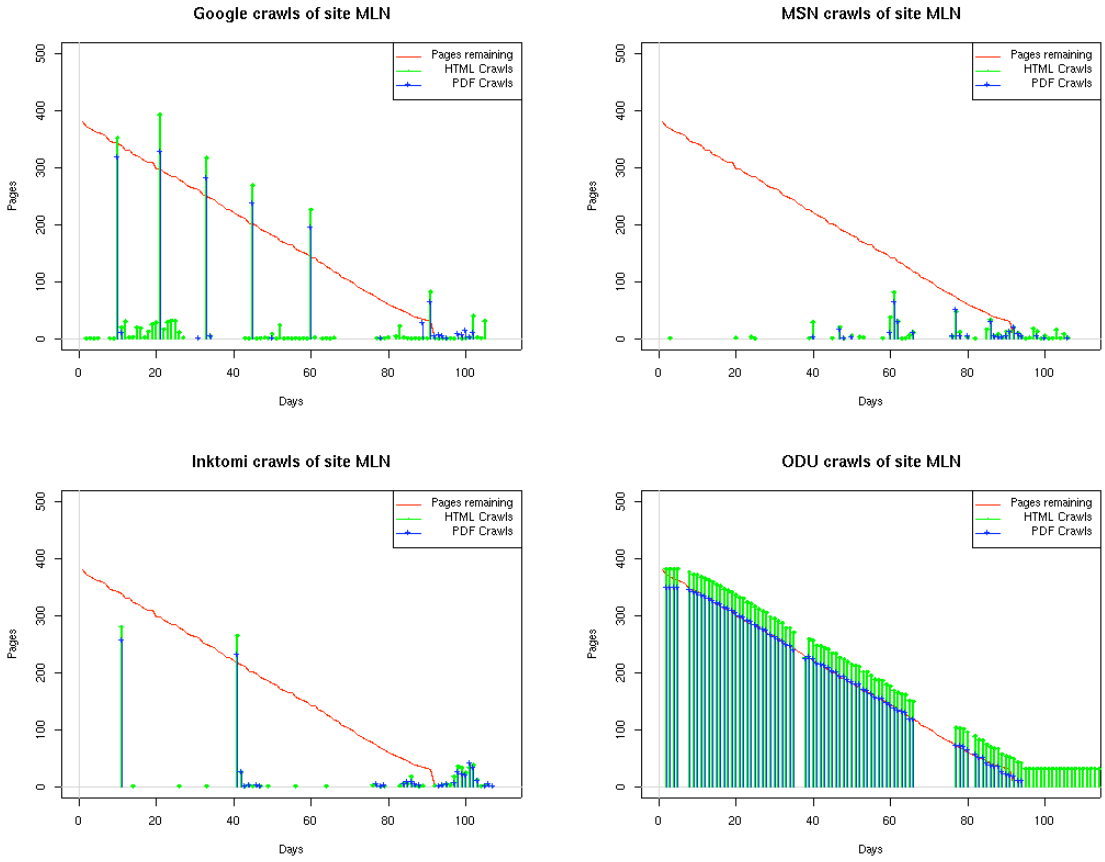


FIG. 22: Crawling patterns on site MLN. Crawling patterns of Google, MSN, Inktomi/Yahoo & the ODU Crawlers on site MLN. The X-axis indicates the experiment day number and thus the frequency of crawler visits, as well as the depth of each crawl. The ODU robot, provided for comparison purposes, was the heaviest user of the site, as the graph shows. The diagonal (red) line indicates the original limit of source content. Robots sometimes attempted to guess URLs that did not exist; see Chapter I, Section 1.4 on page 10 for examples of "guessing". Gaps in the graphs reflect a nearly two-week period during which the web server's logging facility was not operating.

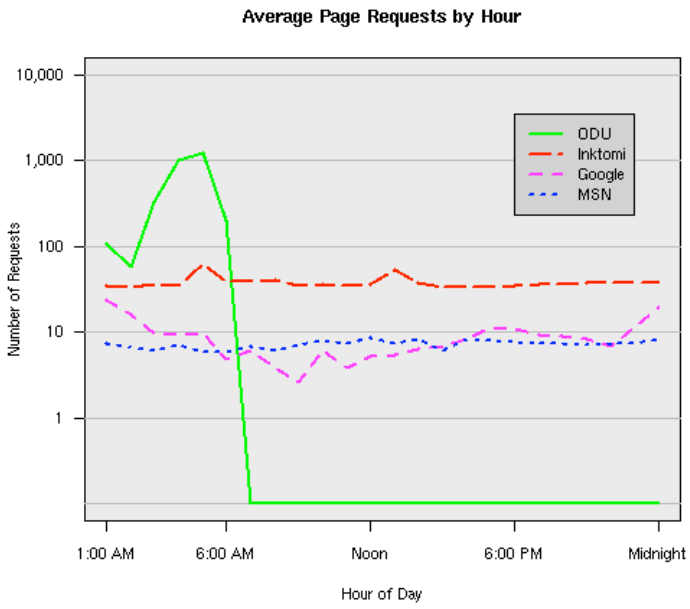


FIG. 23: Hourly crawling patterns on site MLN. Crawling patterns by time of day for Google, MSN, Inktomi/Yahoo & the ODU Crawlers on site MLN. ODU's robot limited its crawls to the early morning hours, whereas the major search engines maintained a relatively steady pace of requests throughout the day.

Experiment Results

The discovery timeline of the websites varied by crawler. Google arrived within 3 days at each of the sites, and MSN within 5 days, while Yahoo (Inktomi) took 2 weeks to visit. The first visit date bears no relation to the crawling pattern, however. Google arrived early and returned often, crawling the sites as thoroughly as the remaining resources allowed. MSN, which had visited early, failed to show much crawling interest. It did not request more than the main page for over one month from the start of the experiment. Yahoo's arrival was latest, but it immediately explored nearly all of the available site. These arrival and crawl depths can be seen for the MLN site in Figure 22.

All of the crawlers showed a preference for HTML resources. Very few of the images were ever crawled, even by image-specialists like PicSearch. PDF files were crawled more than images, but still significantly less often than HTML. As of 2005, then, the search engines appeared to be particularly focused on HTML resources. They were very persistent in repeating requests for HTML resources that had been deleted (see the animations in [170] for a dynamic view of the behavior). These resources also persisted longer in cache [123], making HTML files a better candidate for emergency restoration from cache which could potentially translate into having better preservation prospects.

The breadth of the website had no impact on the crawlers. Arrival time appears to depend on the cycle of the particular web crawler rather than on the site's page rank, for example. Whether or not depth would matter was not clear from this experiment. The author devised a much larger and longer-duration experiment to explore this aspect of crawler behavior, which is discussed in the next section.

1.3 Crawler Behavior on Wide and Deep Websites

A year-long "Deep Website" experiment conducted at multiple sites provided more insight into crawler behavior and limitations [175]. Conventional wisdom holds that search engines "prefer" sites that are wide rather than deep, and that having a site index will result in more thorough crawling by the Big Three crawlers – Google, Yahoo, and MSN. Simply put, the resources on a wide website are found near root level, and have very few "slashes" in the URL: `http://foo.edu/dirI/X.html`, for example. The resources on a deep website occur many levels below root and will have several slashes in the URL: `http://foo.edu/dirA/dirB/dirC/dirD/dirE/dirF/dirG/dirH/dirI/X.html`, for example. Figure 24 illustrates the difference between these two design approaches.

At only 3 levels, the Decaying Directories experiment of Section 1.2 in this chapter did not have sufficient depth to test the willingness of crawlers to explore beyond the topmost levels of a website. The author created a series of live websites, two ".com" sites and two ".edu" sites, that

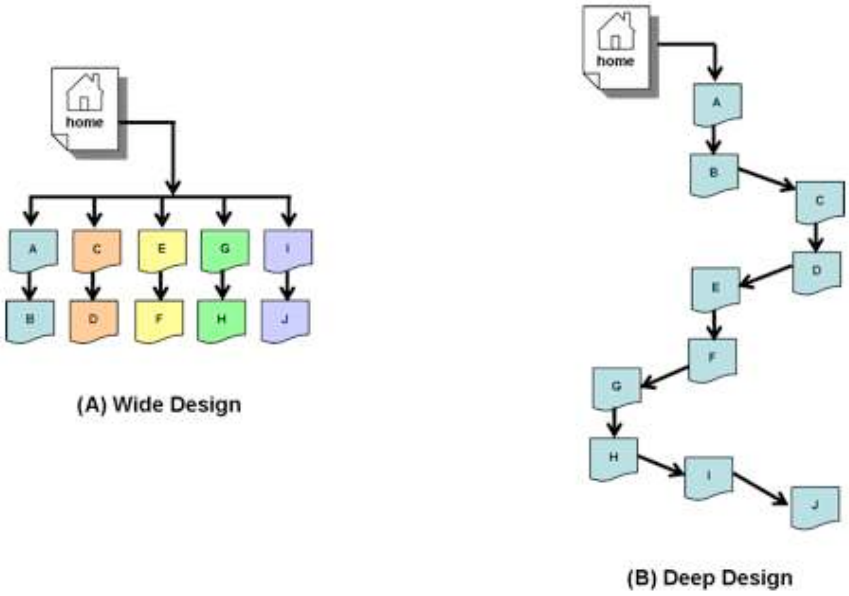


FIG. 24: Wide and deep websites. A wide website has a large number of directories and resources listed at or near root level, i.e., there are only a few "slashes" in the URL. Slashes are an indication of depth in a website. Figure (A) is wide, and its 20 resources (A to J) will each only have one or two slashes. For example, the URL for J.html in (A) would be `http://foo.edu/dirI/J.html`. A deep website has many slashes in its URLs. The URL for J.html in (B) would be `http://foo.edu/dirA/dirB/dirC/dirD/dirE/dirF/dirG/dirH/dirI/J.html` which has 10 slashes, i.e., it is very deep.

TABLE 10: Website content of the "Bread-Crumb" (BC) and "Buffet" (BF) websites. Text content on each of the sites varied because the source texts (English literature) were different. Images and index pages were nearly identical in size across the sites. Size is shown in bytes.

		Avg Size (KB)			
Qty	Description	BF-.com	BC-.com	BF-.odu	BC-.edu
1	Web Root ("Home")	5677	5677	5910	5888
100	Subdirectory Index	47677	1166	47906	1361
20,000	Main Content (HTML)	2325	2295	2599	2624
202	Images (GIF/PNG/JPEG)	11483	11482	4586	4723
20,302	URIs per site	Avg Website Size: 95 MB			

were very wide and very deep to see if structure appeared to impact crawlers.

Part of the goal of the "Deep Website" experiment was to compare crawling tactics employed by the three major crawlers, Google, Yahoo, and MSN: did site design appear to affect crawling patterns? Another goal was to see if the crawlers would explore the full depth and breadth of the sites, which were both very wide (100 directories wide) and very deep (100 directories deep). Would crawlers reach every resource?

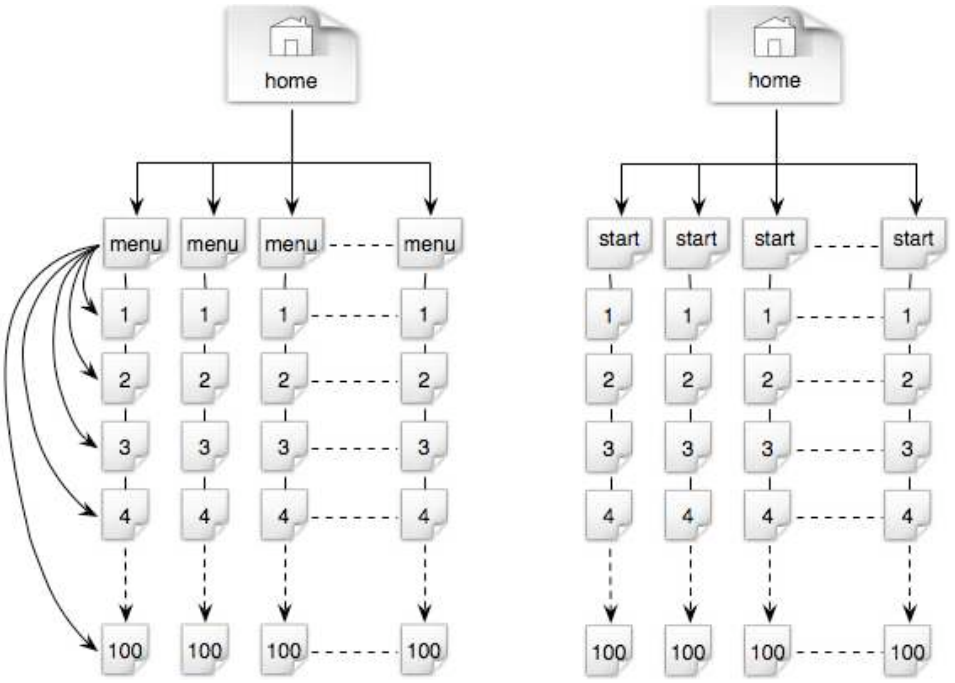
Website Design and Implementation

Reviewers of the Decaying Directories experiment wondered whether the artificial content of those sites influenced crawler behavior. Given the persistent behavior of search engine crawlers on the Decaying Directories sites, the author felt that crawlers were not "aware" of the random content and therefore not affected by it. Nonetheless, when designing the follow-on experiment a different text-building approach was taken for each of the site resources.

An extensive subset of English-language texts was extracted from Project Gutenberg[3] to populate the text portion of the websites. The text was processed where necessary to convert it from UTF-8 or other character set, to plain ASCII. In addition, every text was linked to the appropriate title and author; each page had at least 200 words; and each ended in a complete sentence or poetic stanza. The total number of URIs on each site exceeded 20,000. Figure 25 shows the structure of the website and Table 10 specifies content distribution by file type.

Each site had *unique* content to ensure that no pages would be seen as "mirror sites" since that might impact a crawler's willingness to explore the site further. The author's scripts created a very user-friendly website, complete with links and descriptive information on each page. In sum, the

[3] `http://www.gutenberg.org`



(A) Buffet Website (B) Bread-Crumb Website

FIG. 25: Experiment website schema. Both Buffet (A) and Bread-Crumb (B) websites were very deep, with 100 directories nested below each root-level directory. The main difference between the two sites was their internal linking structure, that is, in the location of links to each of the site's pages.

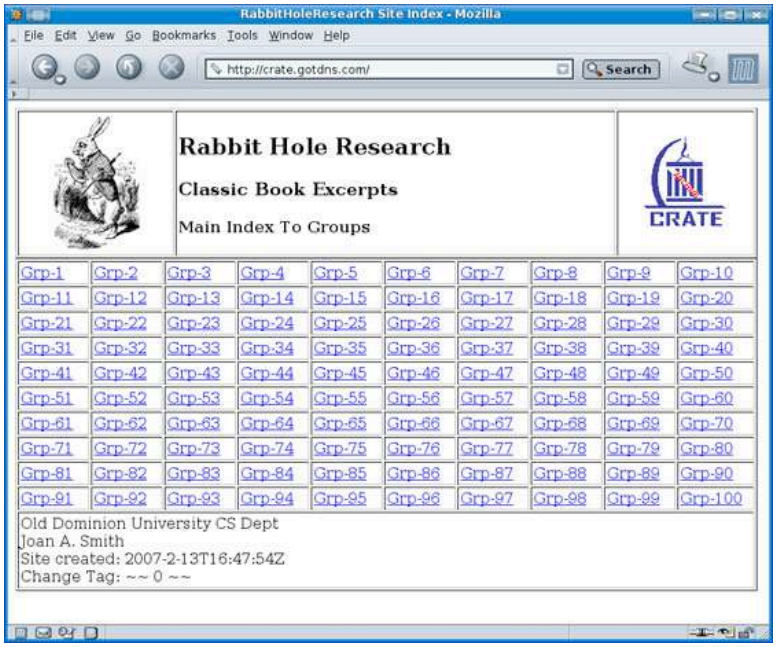


FIG. 26: Experiment website main page (web root). The page provides a link to the top page in each of the 100 subdirectories. This page is shown as "Home" in Figure 25.

sites have the overall look and feel of "real" websites, attested to by the occasional visitor who arrived via a Google search for that author or quotation.

Two types of website were created, one that provided a view into the full site resource set via a series of index pages, and another that forced page-by-page crawls to reach every resource on the site. At root level there were 100 directories called "groups" (g1 through g100). The test site had a single root page (index.html) which contained a link to each of 30 subdirectories, as shown in Figure 26.

Each of these in turn had a set of nested directories called below them that were 100-levels deep (d1 through d100). The paths to the resources could be shallow or deep, i.e., they could have many slashes or only a few (slashes, not text size, indicate depth):

- http://blanche-00.cs.odu.edu/g25/d1/d2/d3/5.html
- http://blanche-00.cs.odu.edu/g13/d1/d2/d3/d4/d5/d6/d7/d8/d9/17.html
- http://blanche-00.cs.odu.edu/g95/d1/d2/d3/d4/d5/d6/d7/d8/d9/d10/. . ./d15/30.html

The two types of sites differed primarily in link organization. The first type, the author termed a "Buffet Site" since it is similar to having everything available and visible in just a few, easily accessed locations like the Salad Bar and Dessert Center at a self-serve restaurant. The other type of site, a "Bread-Crumb Site," only exposes a few new pages at a time. The search engine (or user)

must delve into the site one page at a time in order to discover all of the content. This approach is more like following a trail of bread crumbs through the forest.

Two Buffet sites were installed and two Bread-Crumb sites, one each in the ".com" domain and the ".edu" domain:

1. http://crate.gotdns.com (Buffet.com)
2. http://blanche-00.cs.odu.edu (Buffet.edu)
3. http://oducrate.gotdns.com (Bread-Crumb.com)
4. http://blanche-02.cs.odu.edu (Bread-Crumb.edu)

The physical layout of each website was identical, but the internal resource-linking method differed, having one of two possible structures.

The first structure option has a high-level resource list, as shown in Figure 27. Websites with this structure are called Buffet Sites in the experiments, because they have a "menu" of URLs on a web page at the first level below Root. There is one menu for each of the 100 directories across the width of the website. A crawler or user visiting the site can select any one of the directories and from there find the menu of resources that exist below it. The visitor can then select any of the links and jump directly to that URL, even to the very bottom of the website, which is a directory 100 levels below root.

The second structure option lists only one or two URLs per web page. A visitor follows the same sequence of selecting one of the 100 directories across the width of the website. The page reached from that link has only one URL, which in turn leads to a web page with only a few links on it. To explore the full depth of this directory, the visitor must continue to follow each of the links on subsequent pages. In short, it is similar to following a bread-crumb trail. An example of a Bread-Crumb website displaying only one link on the directory entry page is shown in Figure 28.

The website content was built using a simple HTML template and script. This approach emulates current web practices, as approximately 40–50% of web content is templates [59]. Each site has an average 95 MB of content (Table 10) culled from public domain sources. There is no hidden content, and since the text was extracted from classic literature, it makes sense grammatically and linguistically.

Data Collection

All four sites were installed simultaneously in February 2007. Once a website is published, there is an unpredictable delay before search engine crawlers arrive, even if the website owner directly notifies Google, Yahoo, and MSN. On the other hand, new links on existing websites are crawled relatively soon after first appearance. Therefore, to facilitate website discovery, the sites were advertised on other websites which were already being regularly crawled by Google, Yahoo, and MSN:

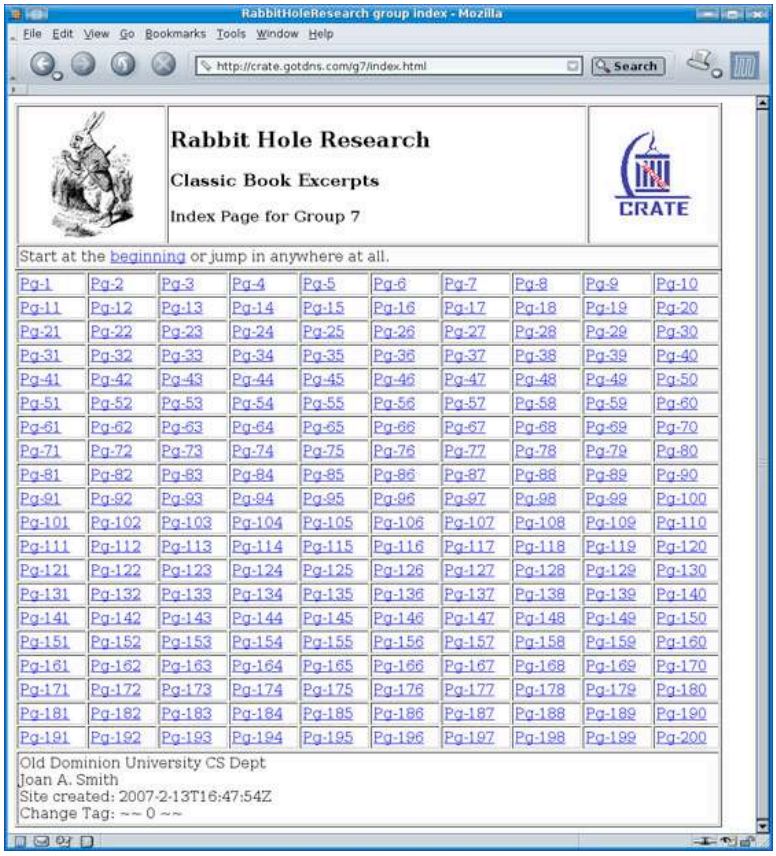


FIG. 27: Buffet website directory entry page. This web page is like a menu of URLs. It provides a direct link to each resource in the 100 directories below it. This page is represented as the top row below Home in Figure 25-(A).

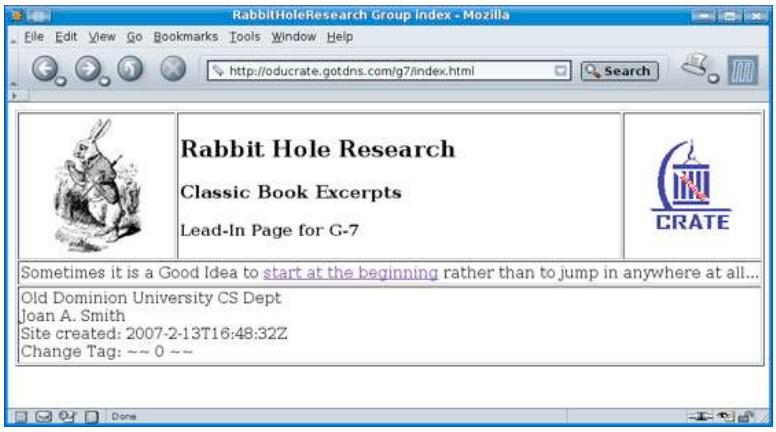


FIG. 28: Bread-Crumb website directory entry page. This type of website forced visitors to traverse every page in order to find all of the website's resources, much like following a bread crumb trail. This page is represented as the top row below Home in Figure 25-(B).

TABLE 11: Deep website experiments: Discovery and completed crawl times.

Site	Data Point	Google	MSN	Yahoo
Buffet.com	Days to Discovery	1	1	2
	Days to Max Crawl	18	213	365
(crate.gotdns.com)	Percent of Site Crawled	100%	100%	44%
Bread-Crumb.com	Days to Discovery	1	1	2
	Days to Max Crawl	44	293	365
(oducrate.gotdns.com)	Percent of Site Crawled	100%	98%	3%
Buffet.edu	Days to Discovery	12	20	14
	Days to Max Crawl	105	245	100
(blanche-00.cs.odu.edu)	Percent of Site Crawled	100%	100%	45%
Bread-Crumb.edu	Days to Discovery	12	18	13
	Days to Max Crawl	298	294	253
(blanche-02.cs.odu.edu)	Percent of Site Crawled	99%	7%	2%

- http://www.cs.odu.edu/∼mln/
- http://www.cs.odu.edu/∼fmccown/
- http://www.cs.odu.edu/∼mklein/
- http://www.cs.odu.edu/∼jsmit/
- http://www.joanasmith.com/
- http://www.owenbrau.com/

"Advertising" in this case simply means that these websites placed a link to each of the 4 experimental websites. On the next visit from Google, Yahoo, or MSN, the links would be found and added to the search engine's list of URLs to visit. Table 11 lists the inception date for each of the experimental websites, the first visit date for each of the search engines, and the date when the first full crawl of each site was completed.

Once each website was installed, its logs were examined daily for crawler activity by Google, Yahoo, and MSN (only). The author's utilities from the Decaying Directories experiment were adapted to collect data from the Deep Website experiment and map it into vectors for graphing. For this experiment, the sites remained static. No resource was moved or changed during the year-long experiment (through March 2008).

A variety of system administration problems occurred during the more than 12 months of the experiment. Most of these affected the .edu sites, reducing the number of consecutive log-days available for analysis from 365 to 289. A short, 2-day availability gap occurred early at the .com sites when the host provider changed IP addresses, but this did not appear to affect the crawlers. Such events are relatively common on the web. In all, more than a million requests by Google, Yahoo, and MSN were processed during the experiment. Table 12 summarizes the

TABLE 12: Deep website experiments: Crawler request statistics

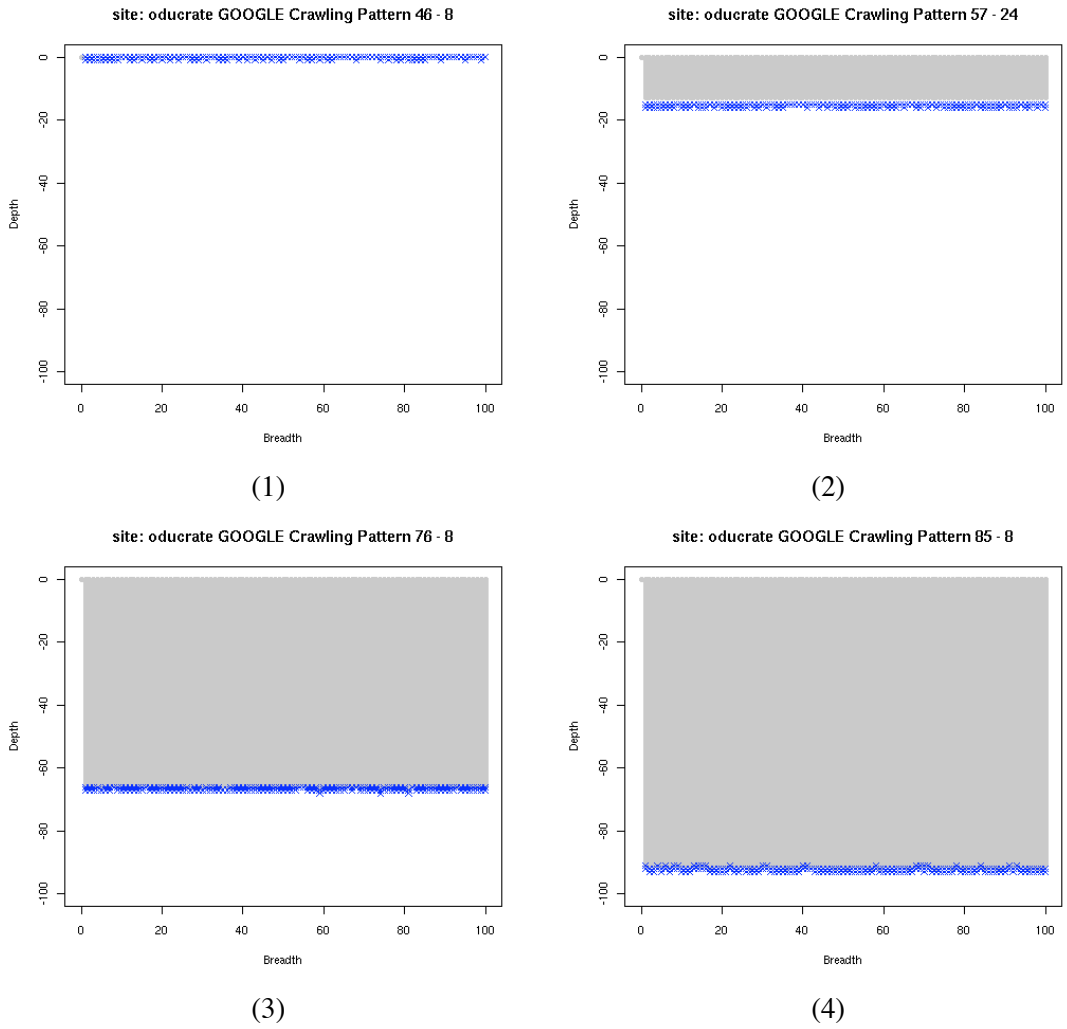
Site	Data Point	Google	MSN	Yahoo
Buffet.com (crate.gotdns.com)	Total Requests	262,851	96,109	50,378
	Avg. Requests/non-index page	13	5	6
	Total Standard Requests	106,073	96,109	40,254
	Total Conditional Requests	156,778	90	10,124
	Total Root-index Requests	88	110	241
	Total Menu-index Requests	1,050	2,712	10,106
Bread-Crumb.com (oducrate.gotdns.com)	Total Requests	41,809	4,296	10,410
	Avg. Requests/non-index page	2	4	7
	Total Standard Requests	34,160	4,267	6,917
	Total Conditional Requests	7,649	29	3,493
	Total Root-index Requests	58	110	206
	Total Menu-index Requests	1,091	2,343	6,021
Buffet.edu (blanche-00.cs.odu.edu)	Total Requests	223,963	276,602	53,735
	Avg. Requests/non-index page	11	14	2
	Total Standard Requests	77,310	276,602	41,896
	Total Conditional Requests	146,653	0	11,839
	Total Root-index Requests	70	101	150
	Total Menu-index Requests	783	2,865	20,175
Bread-Crumb.edu (blanche-02.cs.odu.edu)	Total Requests	34,745	10,703	6,954
	Avg. Requests/non-index page	2	6	6
	Total Standard Requests	29,822	10,703	5,188
	Total Conditional Requests	4,923	0	1,766
	Total Root-index Requests	50	89	125
	Total Menu-index Requests	700	2,990	4,375

request statistics for each website and crawler.

Crawler Characteristics

Each crawler exhibited different access and persistence patterns, and these patterns varied by domain (.com or .edu). In general, width was crawled more thoroughly and quickly than depth. Upper-level URL menu pages like those on the Buffet sites improved crawler penetration. Google was quick to reach and explore the new sites, whereas MSN and Yahoo were slow to arrive, and the percentage of site coverage varied by site structure and by top-level domain. The data for each of these is shown in Table 11.

The logs show some interesting site access patterns by the Google, Yahoo and MSN crawlers. Figure 29 is a series of snapshots of the Googlebot's progress through the Bread-Crumb site, "oducrate.gotdns.com". These are best seen in an animation of the activity, available in [175],



(1) (2)

(3) (4)

FIG. 29: Google's crawling pattern on Bread-Crumb.com website. This shows various stages of the Googlebot's activity on the Bread-Crumb.com site (oducrate.gotdns.com). Note the almost regimented progress through the links. The background grays to indicate links that have been crawled one or more prior days, and the X indicates the active request of the moment. Google and the other search engines have numerous individual robots that may visit a site simultaneously to harvest the data.

where Google's robots advance through the Bread-Crumb .com site like a formation of soldiers on parade. Other robots were not quite as systematic on the Bread-Crumb sites. For the Buffet Sites, the access patterns were considerably more random, as Figure 30 shows. In fact, each robot approached each site differently. For comparison, see Appendix A which presents graphs for each of the three robots at each site.

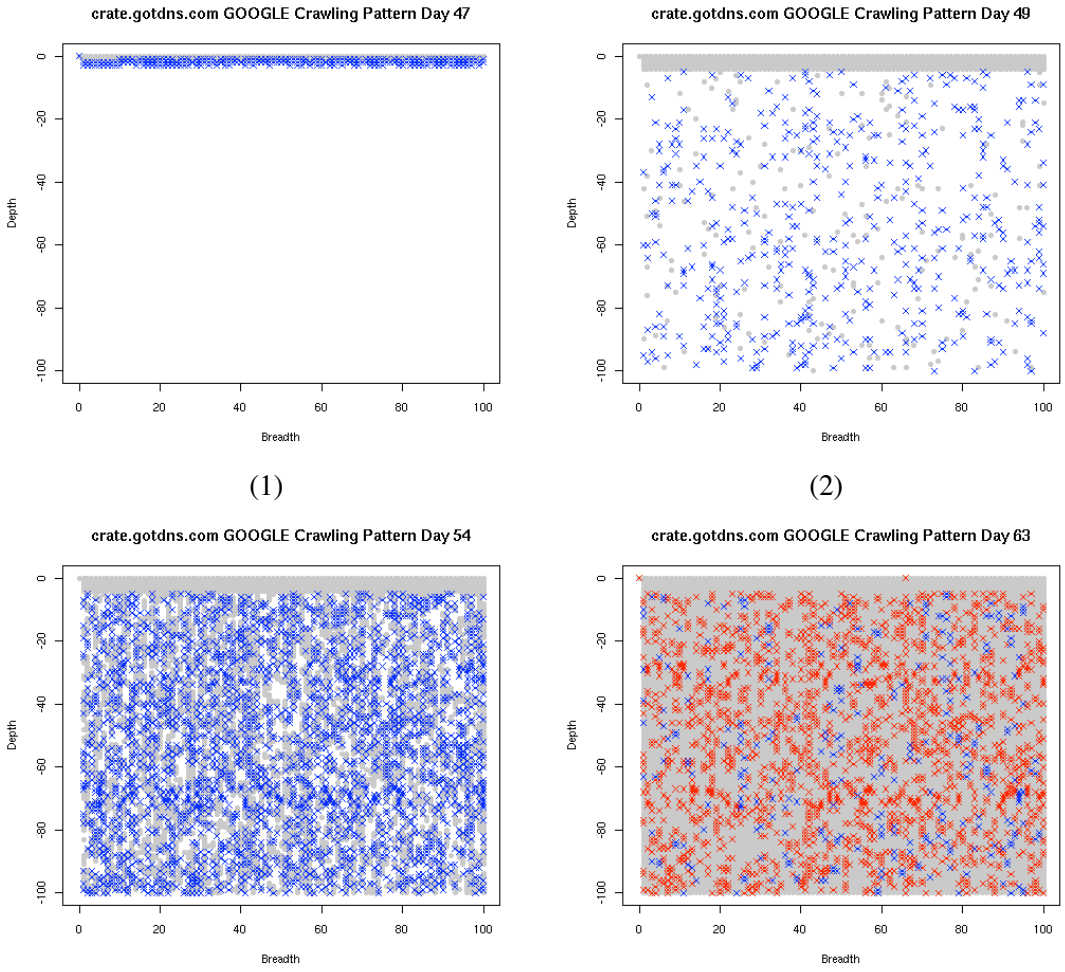
Experiment Results

The design of the site impacted depth, breadth and time that the search engines spent exploring the sites. This difference is clearly seen in Figures 31 and 32. The main concern is the lengthy timetable to reach full crawl status by more than one crawler. Depth and speed to coverage was quite different between the two higher-level domains. The Dot-Com Buffet Site had faster and more complete coverage than the Dot-Edu Buffet Site. The opposite was true for the Bread-Crumb sites. Is this an issue of trust? Sites called "spider traps" [145] certainly exist in the Dot-Com domain, and the continually-linking-down structure of the Bread-Crumb site could be interpreted as such a trap. If search engines believe that this spider-trap activity is not prevalent in the Dot-Edu domain, they might be more willing to explore the Bread-Crumb site more fully there. In any case, preservation requires discovery, ideally by more than one agent; design therefore needs to be a consideration. If data had been removed on a random or even specific schedule (as in the Decaying Directories experiment), many pages on the sites would not have been discovered at all.

1.4 Summary of Crawler Observations

The timeline from initial discovery to in-depth crawling varied for each search engine. MSN appeared to "wait" for a few months before beginning its crawls of the .com websites. Is this an example of a policy or of limited resources available for crawling? A policy, if it existed, might specify a time-delay before exploration because websites appear and disappear at a rapid rate; such a policy could reduce wasted indexing of sites that will not endure. Another question that is as yet unanswered is whether this behavior would be repeated if the experiment were repeated for a new set of websites. If it is a policy, then preservation of short-lived websites is at risk, insofar as they would otherwise be captured in the search engine's cache.

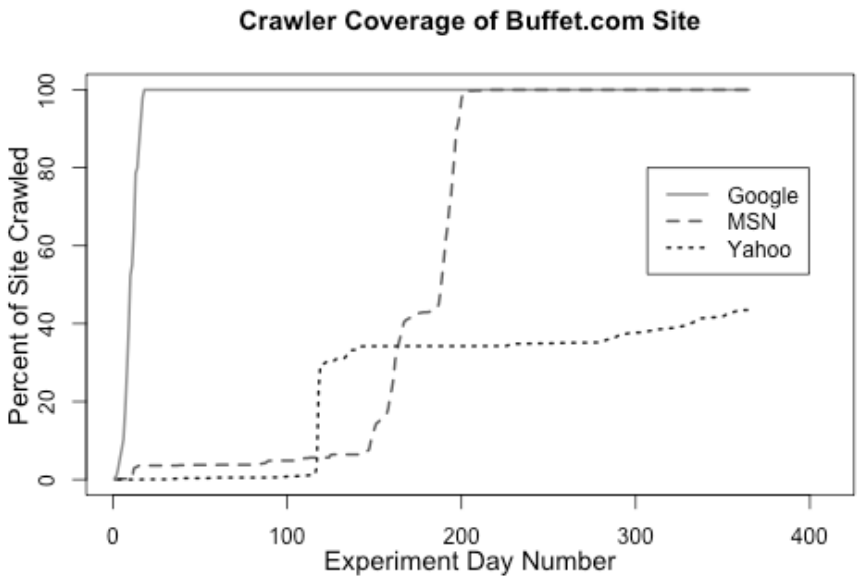
The contrast between Google and Yahoo, and between MSN and Yahoo, is striking. Like MSN, Yahoo exhibits a "preference" for the menu-style design of the Buffet websites, whether .com or .edu is the Top-Level-Domain. Yahoo also delayed months before exploring beyond the first level of the websites. Again, whether this is due to a policy or to limited resource availability is not known. However, Yahoo's request rate was relatively high, as Table 12 illustrates. With over 50,000 requests at the Buffet.com website, it is surprising that its coverage is so low – a mere



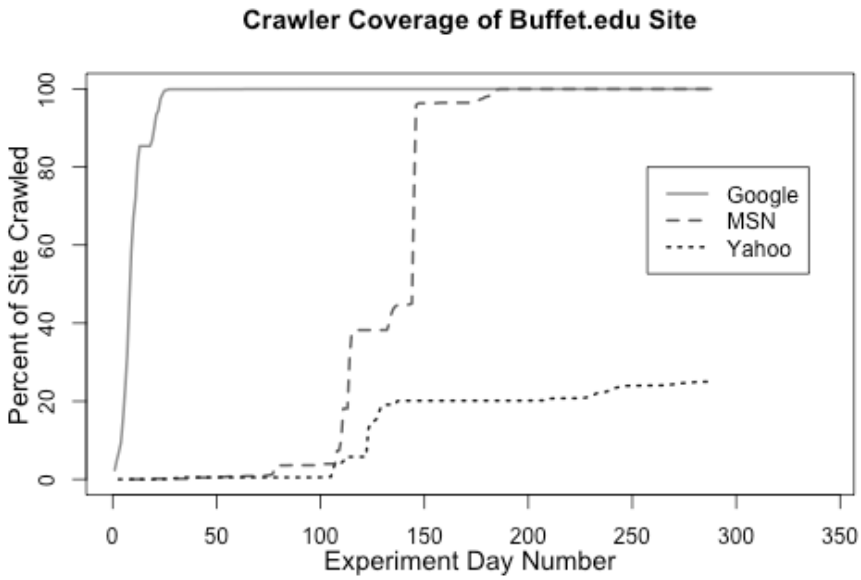
(1) (2)

(3) (4)

FIG. 30: Google's crawling pattern on Buffet.com site. This shows the Google robot's crawling patterns on the Buffet site (crate.gotdns.com), which are very different from the Bread-Crumb site (Figure 29). Here, the robots take advantage of the top-level list of resources and simultaneously harvest pages scattered throughout the site. Google also uses its request patterns judiciously. In (3), Google is still visiting links for the first time. In (4), the Googlebot's requests are primarily *conditional* GETs, with only 10% being a regular, non-conditional GET.

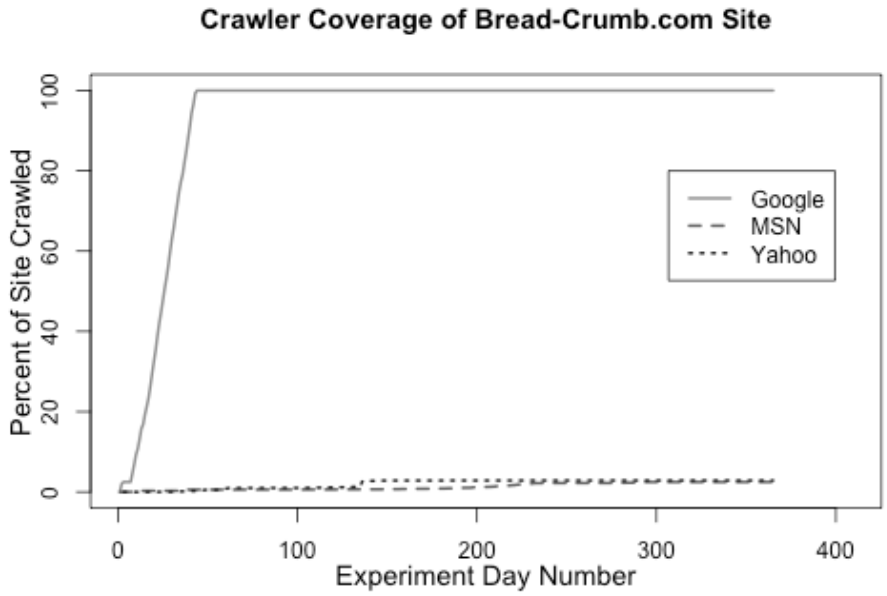


(A) Buffet.com Website

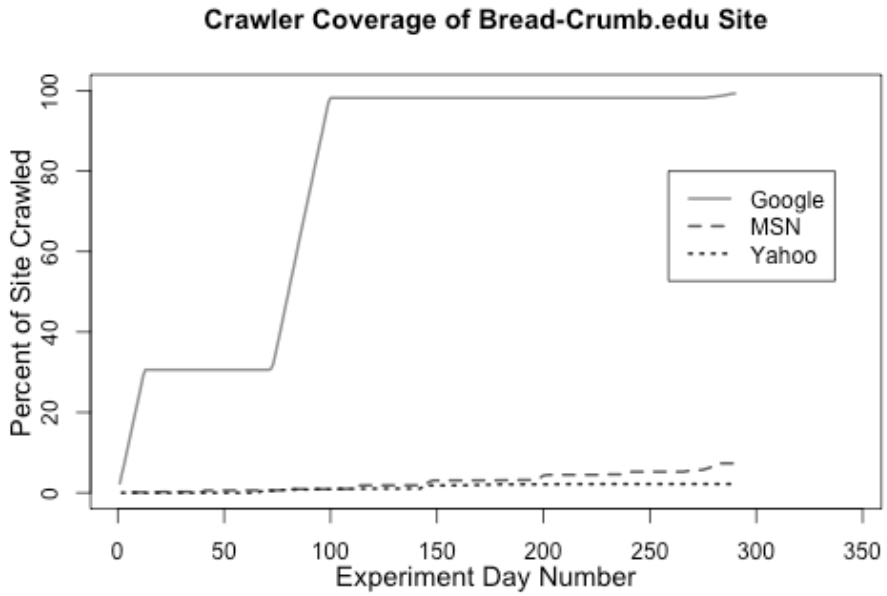


(B) Buffet.edu Website

FIG. 31: Buffet website crawler coverage. This is the coverage accomplished by the robots from Google, Yahoo, and MSN. The percent coverage and time to explore a site appear to be affected by *both* site design and high-level domain. On these sites, every resource is accessible via a high-level set of indexes, or "menu" of the site.



(A) Bread-Crumb.com Site



(B) Bread-Crumb.edu Site

FIG. 32: Bread-Crumb website crawler coverage. Although all of the site's links are accessible, *every page* must be individually crawled to find them all. The crawl coverage graphs shown here differ considerably from those in Figure 31, implying that site design does affect crawlers.

44% of the website. Looking at it from the perspective of coverage/requests, Yahoo is inefficient. Is this intentional or does it represent a poor algorithm design? As with the questions surrounding MSN's behavior, a single experiment is insufficient to determine the answer.

In sum, all of the crawlers behaved differently on each of the websites. Google was the most aggressive across the board, accessing a large number of links every day throughout the experiment regardless of the design of the website, Bread-Crumb or Buffet. The other crawlers exhibit a preference for "wide" rather than "deep" sites, and for sites which provide high-level indexing of site contents. Some search engines are more thorough than others for sites that do not meet this criteria, but preservation is best served by maximum exposure of site resources. Cooperating with the crawlers by providing easily-accessed content can improve the website's likelihood of exposing all of its resources for general accessibility and future preservation.

2 THE SEARCH ENGINE AS AGENT OF REFRESHING

An important preservation task is *refreshing* the bits. Search engines assist with this task through their continuous crawls of websites, which are then accessible to the general public. A user could, at any point, store a copy of all or part of a website thus acting as an incidental preservation agent. In the meantime, the search engine continues to revisit the site and update or validate its record of the site. The refreshing of bits continues for as long as the site is available.

One aspect that can affect refreshing is the frequency that the crawler visits the website. The animated views of search engine activity on the experimental sites in [170] and [175] show an apparent change in the access frequency of crawlers between the first Decaying Directories experiment (2005) and the Deep Website experiment (2007-2008). All three have improved their speed and depth of coverage, but Google still has better coverage metrics. In particular, Google is more constant and methodical when refreshing its crawl of a site. Yahoo and MSN are both less thorough, less methodical, and take much longer to fully refresh whatever portion of the site they did visit. The top-most levels appear to be refreshed very frequently, but that is not the case for levels below that. Google may be better than the others, but for preservation purposes websites need improved coverage and refreshing by all of the search engines.

Another difference between Google and the other search engines is in the use of the conditional request, i.e., requesting the resource only if it is newer than the timestamp of Google's last visit (Google supplies that timestamp). Both Yahoo and MSN use this feature rarely; each time they request a resource it is as though they have never seen it. In this respect, Google appears to be relying on its cache, whereas MSN and Yahoo are perhaps refreshing their copies of the resource with each visit. Whatever the purpose, Google is "friendlier" than MSN and Yahoo, because a conditional request puts less load on the web server, which needs only to reply "304 Not Modified" instead of responding with the full resource file. For dynamic resources, this advantage will vary

with the specific implementation of content handling by the web server.

3 THE SEARCH ENGINE AS AGENT OF PRESERVATION

3.1 Enumerating Website Resources

Search engines are used in the expectation that they can find relevant content. In turn, this implies that they can enumerate the resources available from websites. They build their content lists by crawling websites, but website design may hinder search engine access or prove incompatible with the search engine's policies or robot availability. Experiment #1 (Decaying Websites) and experiment #2 (Buffet & Bread-Crumb) showed that search engines may have problems building the list of website resources if (a) they are too short-lived on the website and (b) if access requires following a long series of embedded links scattered throughout a website. To fully "count" or enumerate a website's resources, the website structure needs to support the crawling process. Crawled resources may be *found* resources, increasing the likelihood of preservation by anonymous third-party clients who retain copies of information they find useful or otherwise worthwhile.

3.2 The Web Infrastructure

Beyond simply revisiting and refreshing the bits of data, search engines also keep copies of many of the pages that they visit. This cache, or *web infrastructure*, is accessible even if the source site is temporarily off-line. In some cases, the cache contains an exact duplicate (this is particularly true of HTML pages). In other cases, the resource is a reconstructed version, for "standard browsability." That is, if the resource format typically requires a special browser plugin or third-party program (MS Word, Acrobat Reader), the search engine may convert the content into text and possibly images so that it is viewable in the search engine's cache without visiting the original site itself. Outside of the competitive benefits it confers on the search engine, the existence of this cache and a client's ability to examine its contents adds another dimension to the search engine's role as an agent of preservation.

3.3 Lazy Preservation

The term *Lazy Preservation* was coined by Michael Nelson, and adopted by Frank McCown and the author [134, 123] to describe the ability to recover websites from the web infrastructure. In some cases, a complete website could be reconstructed using only the content from search engine caches [96]. A set of tools developed by McCown [119, 120] has been used by numerous sites worldwide to restore lost websites. The web infrastructure can act both as an emergency backup and as a true preservation source, since it refreshes and migrates content continually. The ability to restore something that no longer exists in its own right is closely associated with the intent of

preservation. Search engines are therefore important to preservation on many levels ranging from discovery to recovery.

4 SUMMARY

Search engines and archives have several goals in common, including content freshness, site coverage, and accessibility. But the focus of a search engine is today, whereas the focus of an archive is both past and future. Information must be found to be preserved, and search engines contribute by making information *discoverable*. In some cases search engines help by converting or migrating resources to another format, if only for display simplicity. Search engines are constrained by their ability to locate and crawl web content, which puts much of the burden of discovery upon the webmaster or web designer to facilitate this process. Search engines need help "counting" (finding) site resources, but will often keep a cached copy of those resources that they do find. They may refresh their copies with each visit or instead rely on the integrity of the cached copy. In sum, search engines act as agents of discovery, refreshing, and near-term preservation for websites.

CHAPTER V

RESOURCE ENUMERATION: THE COUNTING PROBLEM

In these matters the only certainty is that nothing is certain.
Pliny the Elder (23 AD – 79 AD)

1 THE COUNTING PROBLEM DEFINED

Websites consist of web resources which may originate from files, dynamic content, or a combination of different types of content. As discussed in the Introduction (Section 1.3 on page 4), a website (W) is *a collection of resources having a common domain name, accessible via the Internet*. A web resource (w), also defined in the Introduction, is *the representation returned by the server in response to dereferencing a URI*. More simply:

$$W = \{w_1, w_2, \cdots, w_n\} \tag{7}$$

This is true whether or not the members of W are discreet entities on a file system; and whether or not they combine to form new resources that exist only upon delivery to the client; and whether or not the file system contains entities that are not part of the website. (The links that may exist between members of W are not considered, only the individual resource itself). That is, the mapping from file system to resource may be one-to-one, many-to-one, one-to-many, or some combination thereof. Regardless of the mapping of a w to a virtual or actual file system resource, website preservation expects a list of each $w \in W$ so that all the web resources which make up website W are identified and preserved. This is the basic counting problem: How to confidently enumerate each member of the set W, i.e., each resource of the website.

1.1 Limitations of File System Mapping

Traditional Digital Libraries (DLs) are deterministic with respect to the number of records held. Most DL repository implementations are accessing a database in which all possible records are knowable. However, web harvesting is different. Consider the following, where W is defined as above, i.e., the set of all possible web resources for a particular web server, and F is defined as the set of files (individually, f) on the file system that the web server can see. Additionally, a dynamic-generation script, d is defined which may create new content using resources outside of web space (not in F). When a client requests w, Apache may map this request in any of several

different ways:

$$w \to f \tag{8}$$

$$w \to d \tag{9}$$

$$w \to (f_x + f_y) \tag{10}$$

$$w \to d(f_x + f_t) \tag{11}$$

Similarly, the File System does not necessarily map directly from f to w. The existence of a resource on the file system does not mean that a direct request for that resource will produce a *web resource*[1]. If the server at `foo.edu` contains a file system resource named `/var/www/Xfoo.html`, it does not guarantee that a request for `http://www.foo.edu/Xfoo.html` will return that resource (nor that it will return any "200 OK" response).

$$f_x \to w_x \tag{12}$$

$$f_z \nrightarrow w_z \tag{13}$$

Neither function is 1-to-1 nor onto. It is simple to check whether a single w maps to f, but given F it is not trivial to generate W. In part, this is due to the virtually infinite ways in which a particular web server can be configured by the webmaster. Symbolic links, access restrictions, and other features available in a web server make the mapping process very difficult.

One problem is that the web server can "cover up" legitimate files. Consider two files, A and B, on a web server. Now consider a web server configuration file with these directives:

```
Alias /A /usr/local/web/htdocs/B
Alias /B /usr/local/web/htdocs/A
```

The resource obtained by web crawling and the resource obtained by looking at the file system will be in contradiction. That is, a visitor requesting `http://www.foo.edu/B` will actually receive the resource from `htdocs/A`, and vice-versa. A user on the file system is unaware of the alias (which exists within the web server, not as a file system link). That user will see `htdocs/B` as if the web server directive did not exist.

Another feature of web servers is the "Location" directive, which defines how to map a client request to server resources, whether file-system based or other. Files can also be covered up by such Location directives and so not be accessible directly from the web server. Automounting of Network File System (NFS) directories is another example of a complication to

[1]This is slightly different from the W3C use of "resource" and "web resource". See [188], where a resource is defined as "anything that has identity" and a web resource is defined as a "resource, identified by a URI, that is a member of the Web Core".

the counting problem. NFS mounted directories shared between many departmental machines is a common deployment scenario and makes it extremely difficult to include UserDir files (e.g., `http://www.cs.odu.edu/~mln/`) since there is no (easy) way to know in advance all possible users. However, these files constitute the majority of files accessible from a web server in a shared-user environment such as a university department.

1.2 File System and Web Server Security

Users and administrators have developed bad habits based on "security through obscurity" – if there is no URL on a web page linking to a resource, a search engine cannot find it. URL guessing (discussed in Chapter I, Section 1.4 on page 10) can reveal previously unknown URLs to search engines. This is similar to the problem of search engines revealing files and metadata that users did not realize were available (e.g., [69, 95, 177]). The preferred method to secure web-accessible files is to require a Username/Password combination (via .htaccess, for example) or to encrypt the files. Other methods, of course, are available such as client-side certificates.

1.3 Hidden Files

Somewhat related to the previous discussion on security is the problem that some web servers, including Apache, will advertise files that it cannot read. For example, a file can be seen in a directory listing, but if the permissions on the file are "000" then no one can actually read the file. The file is listed by Apache since its existence is actually a property of the parent directory, not the file itself.

The Apache web server also uses the IndexIgnore directive to specify patterns for file names that should not be included in a directory listing (e.g., "foo~" "foo#" and other file version conventions). However, if requested directly, Apache will serve it if it is accessible: `http://www.foo.edu/index2.html~`, for example. The Apache semantics in this scenario are similar to "hidden" files in the Unix file system. This has serious implications for the counting problem, particularly where a search engine uses a directory listing to build its view of the website. In this case, more files are available via direct URL than is evident from the directory list. Such semantics can be hidden from the web developer who may not be the person administering the website and so unaware of the impact such directives have on web resource exposure to search engines and other clients.

2 WHY THE COUNTING PROBLEM EXISTS

The ability to find *anything* on the web does not mean *everything* can be found on the web, nor that *something* in particular can be found, even if it does in fact exist on the web in an accessible

location. Site content is typically indexed either *externally* by crawlers which follow links found on websites; or *internally* by sites which provide specific lists of site resources. Both strategies are link-based, in that the internally-generated list is simply a more efficient and – ideally – more complete index of the site's links. There is still uncertainty about how completely the links cover the site: some, most, all? There is no mechanism that can definitively answer that question. This inability to confidently enumerate all of a site's resources is why there is a counting problem.

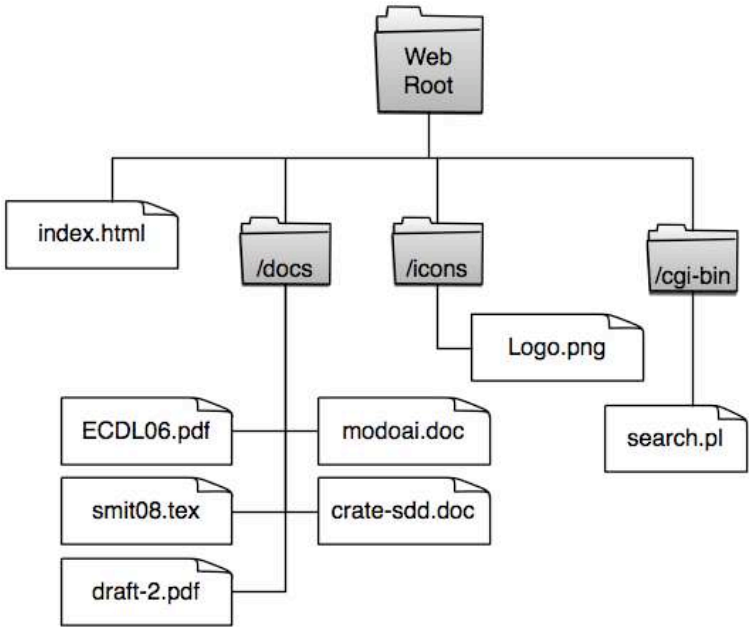
One reason the problem exists is that website file space does not always map directly to website URLs. Figure 33 shows how different parts of a website intersect with the website's host file system. Another reason that the problem exists lies with the underlying protocol, HTTP, and with the limitations of crawling related to that. These two issues, HTTP and crawling, are explored in more detail in the following two sections.

2.1 The Limitations of HTTP

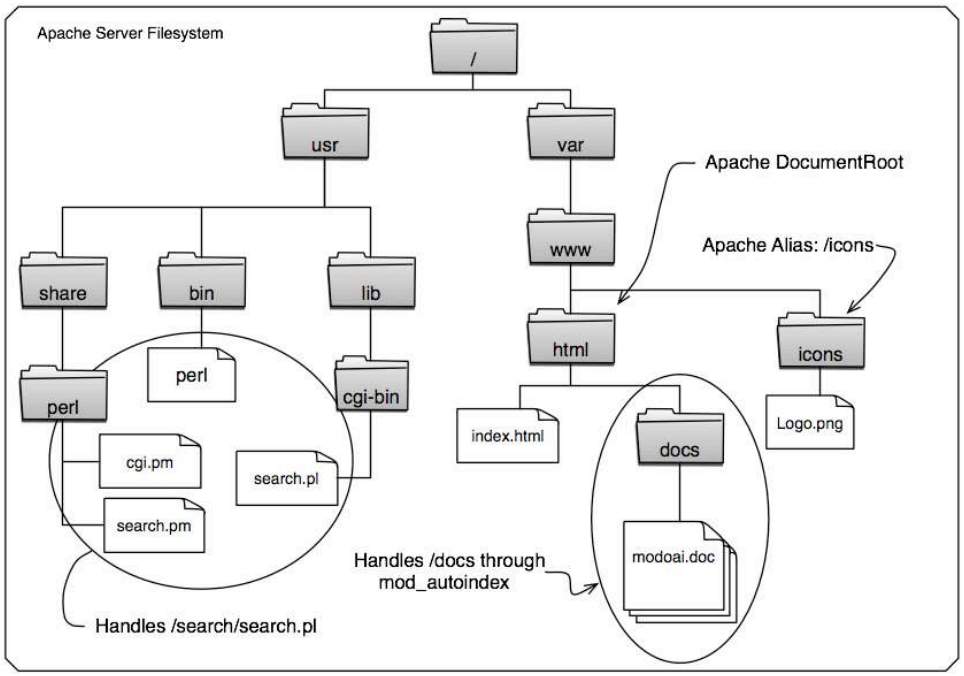
The HyperText Transfer Protocol, HTTP, is the basis for most website-client interaction, whether that client is a *browser* (Internet Explorer, Firefox, Safari) or a *crawler* (Googlebot, Yahoo!Slurp, msnbot). In its minimal invocation (HTTP 1.1), initiating a request is simple:

1. Open a connection to the server
2. Send the 3 parts of the request –
   (a) specify the *method* (GET)
   (b) name the *resource* (file/path/name.html)
   (c) declare the *protocol/version* used (HTTP/1.1)
   (d) specify the host *Host Header* (Host: www.foo.edu)
3. End the request (use Carriage-Return, Line-Feed)

A response can be similarly minimal: `HTTP/1.1 404 Not Found`, consisting of the protocol/version (HTTP/1.1), the response code (in this case, 404), and a *reason phrase* explaining the response code (here, "Not Found"). Although HTTP defines many different *methods*, including GET, POST, PUT, and OPTIONS, HTTP does not have any query or search function (compared with, for example, SQL). The protocol is designed to request a method on a single resource and to get a response to that request. HTTP 1.1 has more headers than those shown in this simple example. Responses can provide more information in the form of headers such as *Content-Length* (size of response in bytes) and *Content-Type* (MIME type of the resource). As helpful and varied as the optional headers are, there is still no header for the client to ask for "wherever name.html is now stored"; the "404" response in the example has left us without the requested resource.



(A) Website Map



(B) File System Map to Website

FIG. 33: A website and its Apache server file system. The file system and the website are not identical. Resources may be aliased from other directories, or called by scripts which live outside of the web root. Mapping from the file system to the website URLs is not a one-to-one process.

2.2 The Limitations of Crawling

Much of the web's usability depends on the efficiency of search engines and their crawlers. The indexable "surface" web has grown from about 200 million pages in 1997 to over 11 billion pages in 2005 [73], and the "deep web" is estimated to be 550 times larger [16]. Considerable attention has therefore been given to increasing the efficiency and scope of web crawlers. A number of techniques to more accurately estimate web page creation and updates [38, 141] and to improve crawling strategies [37, 39] have been proposed. Techniques such as probing search engines with keyword queries and extracting the results are used to increase the scope of web crawls and obtain more of the deep web [32, 87, 109, 143, 150]. Extending the scope of a web crawl has implications on the coverage of search engines and in web preservation [81, 110].

These tactics are necessary because web servers do not have the capability to answer questions of the form "what resources do you have?" and "what resources have changed since 2004-12-27?" A number of approaches have been suggested to add update semantics to HTTP servers, including conventions about how to store indexes as well-known URLs for crawlers [29], and a combination of indexes and HTTP extensions [185]. WebDAV [61] provides some update semantics through HTTP extensions, but has yet to find wide-spread adoption. The RSS syndication formats [152] are widely implemented, but they are designed to expose "new" content rather than a complete set of site resources. Some search engines, notably Google and MSN [65, 198], have taken advantage of sites that operate the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), but they do not provide an open-source, broadly applicable solution. MSN, for example, merely states it is committed to supporting industry standard protocols, of which OAI-PMH is one [198]. Sites with OAI-PMH servers were able to register with MSN's "AcademicLive" search service for enhanced content harvesting, but since MSN has merged AcademicLive with MSN Live [2] OAI-PMH registration and support appear to have been discontinued. Google has also recently announced its intention to drop support for the protocol [128].

The top three search engines, Google, Yahoo and MSN, recently agreed to support the *Sitemap protocol* [167], designed to provide search engine crawlers with a list of available resources. These search engines also let webmasters register their sites, a short-cut to eventual discovery by Domain Name Server update notifications, for example. However, registering sites and creating Sitemaps do not replace the crawling process. Search engines do not take web content descriptions for granted, but process each page that they crawl before it will show up in search results. They also conduct their own crawls of other sites which show up during crawls, the "discovery" method the author used for the Decaying Directories and Deep Websites experiments. Although many optimization techniques have been proposed for efficiently building web resource lists [37, 80], the task is still processor- and time-intensive.

[2] http://blogs.msdn.com/livesearch/archive/2008/05/23/book-search-winding-down.aspx

FIG. 34: HTTP request-response example

```
Request:   GET /index.html HTTP/1.1
           Host:  www.modoai.org
           Accept-Language:  en-us, en;q=0.5
           Accept-Encoding:  gzip, deflate
           If-Modified-Since:  Sun, 22 Oct 2006 08:00:00 GMT
           If-None-Match:  “15b9b090-152c-51c72700”

Response:  HTTP/1.1 304 Not Modified
           Date:  Thu, 09 Nov 2006 21:44:35 GMT
           Server:  Apache/2.2.0
           Connection:  Keep-Alive
           Keep-Alive:  timeout=15, max=100
           Etag:  “15b9b090-152c-51c72700”
```

One of the reasons that crawling is not efficient is that web resources are accessed by following links, typically starting from the web root. Each requested URL may contain a list of other URLs (links), which are appended to a list of resources to be requested by the crawler. The queue of pages to be crawled is thus built from the seed page. Super-efficient crawlers, like Google and Yahoo!, can split the crawling task list among many servers and aggregate the results later. The author noted many servers from both of these search engines acting in parallel during the web crawling experiments described in [170]. Crawlers also keep track of the pages they have visited before. On subsequent crawls, they can issue a status request and choose to not update a page if the returned status is “304” (not modified). Table 34 presents a transcript from an actual response-request sequence.

This can be markedly less efficient than an equivalent request for a list of pages that *have* been modified since that date. If a website consists of 100 pages, then the crawler would have to issue 100 requests asking it if page N was modified, followed by a request for the modified page(s). Typical websites do not support a request for a list of pages that have been modified since a particular date, but this would clearly be significantly more efficient for both server and crawler. Figure 35 shows a typical website and the parts of it that are both crawlable and inaccessible. It is true that some crawlers will not respect the “robots.txt” file which asks them to *not* crawl selected pages, but even badly-behaved crawlers may not be able to find unadvertised links.

Various attempts to “SQL-ize” the crawling process have been made by the IETF and other research teams, but getting new protocols launched can be difficult. The DASL was greeted enthusiastically in 1998 and had all but disappeared by 2002 [46]. DASL’s goal was a lightweight

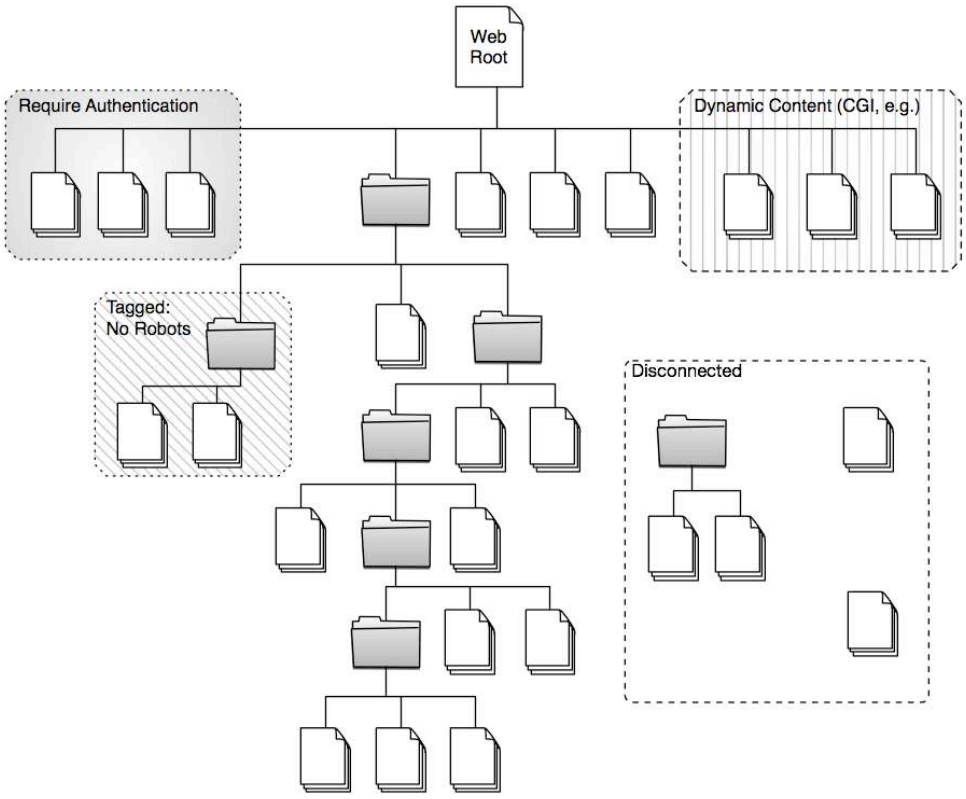


FIG. 35: Crawler's view of a website. The crawler does not "see" any of the content within the dashed box areas. The robots.txt file tells the crawler not to visit certain pages, even though they are accessible. Other pages require user authentication, and still others are dynamically generated (CGI, e.g.). A portion of the site may not be linked internally at all. Some of the latter resources *could* be found if they were linked from external sites.

server-side query capability which would enable clients to search for specific data directly on the remote site server. It specified a minimum basic search grammar that added verbs to HTTP like "search" and "propfind." The verbs are placed directly into the request string, similar to the OAI-PMH examples given earlier. It has been revived recently, but it is too early to know if it will be widely adopted or not.

Another approach is the Harvest Software System, with its indexing tools and "Essence" summarizing algorithm [26]. Harvest also communicates over HTTP and can extract a variety of metadata from a URL, even if the URL is a compressed tar file or a PDF document. Participating sites run "gatherers" which can perform sophisticated tasks like content summary and index generation. Manually-generated metadata can also be incorporated by the tools. Once the resources have been gathered and analyzed, an overall index of the collection is built which can be accessed from a web browser, for example. In addition to gatherers, there are "brokers" who talk to "gatherers" as well as (or instead of) talking to sites directly; the idea is similar to the OAI-PMH "aggregator" mentioned earlier. Harvest has only had limited success, in large part because the individual development team members transferred to "start-ups" during the dot-com boom. The software never reached full maturity, and several of the dependent libraries are no longer available even though the Harvest source code can be downloaded from SourceForge [76]. The author made several attempts on various Linux platforms to install a working Harvest system, but the required libraries either could not be found at all or the versions available were not compatible with the Harvest source code. Attempts to setup a stand-alone Essence extraction system were equally unsuccessful and for the same reasons.

The crawling problem has produced recommendations for web servers on how to improve their "crawlability", and on the effective use of "robots.txt" files to prevent or restrict crawling [29, 149] (to either protect documents or to spare the server needless processing cycles). A crawler which issues requests in the form "has resource X changed?" does not reduce the number of queries that must be issued; it merely reduces the number of pages that must ultimately be refreshed via a new crawl. The author's research showed a fairly high percentage of this type of request across all of the experimental websites [170]. This behavior implies that crawlers expect the ratio of updates to be low relative to the total number of conditional requests issued, although if the web resource has changed, the server will respond with "200 OK" and including the modified resource.

3 THE SITEMAP PROTOCOL

To date, much of the research in the web community has focused on efficiently estimating the updates (i.e., resource changes), deletions and additions of remote, uncooperative web servers. This has been a challenge for all web crawlers, including Google. Now, there is interest in shifting some (if not most) of the responsibility for resource discovery to the web servers themselves, via

```
<urlset>
   <url>
      <loc>http://www.foo.edu/index.html</loc>
      <lastmod>2008-05-22T14:25:21Z</lastmod>
      <priority>1.000</priority>
      <changefreq>weekly</changefreq>
   </url>
   <url>
      <loc>http://www.foo.edu/images/foo2.jpg</loc>
      <lastmod>2008-01-01T04:05:01Z</lastmod>
      <priority>0.600</priority>
      <changefreq>rarely</changefreq>
   </url>
</urlset>
```

FIG. 36: Website URLs in a Sitemap file

the *Sitemap protocol* [167]. Technically a file format rather than a true protocol, it was originally developed and promoted by Google [146, 65] as a means for webmasters to provide a listing of crawlable site resources. Although Google had supported OAI-PMH, and a webmaster could submit a *baseURL*, few websites took advantage of it. The Sitemap protocol specifies the contents and organization of an eponymously-named XML-format file. The file contains a unique resource list for the website. In minimal format, it must contain the following:

(1) XML declaration line

(2) The `<urlset>` tag with XML namespace location attribute

(3) URL information tag set `<url> </url>` (one set per URL)

(4) One URL location tag set `<loc> </loc>` within the URL information tags

(5) The closing `</urlset>` tag.

The file is expected to be located at web root, as, for example, `http://www.foo.edu/Sitemap.xml`. Additional tags are recommend but are not required. They include information helpful to crawlers such as how frequently the resource is updated, and the last modification date. Figure 36 gives a brief example of the urlset section of a Sitemap file. Appendix F discusses the protocol in more detail and provides an example of a complete Sitemap file.

Despite the specific guidelines and many tools for building Sitemaps, it does not solve the Counting Problem. People, tools, and websites are not error-free and a Sitemap will inherit the errors of any and all of these. Some live, automated-update sitemap utilities exist, but these are few and not widely installed. Perhaps the best current example comes from a set of APIs for Apple's latest operating system, OS X. The "web space" portion of the file system maintains a dynamic

sitemap, automatically updating the sitemap as files are added, deleted, or changed. While many dynamic sitemap tools exist, there are none at this point that maintain the synchronization found in OS X. Very few web servers are running on Apple's operating system [139] at this point in time. A particular website could have a perfect Sitemap, but it seems unlikely that this would be the case for most websites.

The Sitemap is a custom file, specific to a particular web server and website; any change to the website must be reflected by a corresponding change to the Sitemap file. The two events do not happen in tandem but in sequence; crawlers could end up requesting non-existent resources or failing to crawl new resources. In sum, there is generally a *race condition* between link updates and crawls, allowing for many opportunities to have site and crawler out of synchronization. At best, the website can make a best effort at providing an up-to-date list of resources with varying degrees of success depending on site content and frequency of change.

4 SUMMARY

The Counting Problem is more accurately described as the website resource enumeration problem: It is difficult, perhaps even impossible, to confidently list all of a website's resources. Even with the definition of website resource restricted to network-accessible web resources, the problem is not trivial to solve. Enumerating all of a site's resources is a challenge for both crawler and webmaster, in part because there is no precise mapping from file system to web resource list. The HTTP protocol is unable to assist directly, because it does not provide a method for asking a site to "give me everything you've got." Website links, the common method for resource discovery, may not point to all of a site's resources, or may point to resources that no longer exist or have been moved or renamed. The existence of a link does not guarantee the existence of a web resource. Sitemaps, created locally by the webmaster (usually) to list all of the website's available resources, are subject to the race condition between site changes and Sitemap file updates. In short, the counting problem is not solved by these approaches (file system traversal, Sitemap creation, and crawling), it is only ameliorated.

CHAPTER VI

EVALUATION OF RESOURCE ENUMERATION METHODS

We used to think that if we knew one, we knew two, because one and one are two.
We are finding that we must learn a great deal more about 'and'.
Sir Arthur Eddington [115]

1 A COUNTING PROBLEM EXPERIMENT

1.1 The CS Department Website Snapshot

Live websites have characteristics that can be hard to duplicate in a test environment where the website is often generated by a script. For example, humans are prone to make typographic errors when making links: on *Page A.html* a link that should point to *Page B.html* instead was mistyped as *Page b.html*, which does not exist since the underlying file system is case-sensitive when it comes to resource names. Websites also experience outages, periods when logging fails, files overwritten accidentally with older copies, and a host of other events that make live websites a more realistic place to test ideas about websites themselves.

To examine the Counting Problem more closely, and to evaluate solutions for developing a list of all canonical URIs at a given site (i.e., a Sitemap), the author obtained an archived copy of the ODU Computer Science Department (CS) website. A snapshot of the CS website was provided for this experiment which was recovered from a backup tape by the website's system administrators. The choice of snapshot was limited by the availability of logs files corresponding to the snapshot timeframe and by the recoverability of the files in the snapshot. Backups are not always reliable, being subject to media decay and other forms of data corruption and physical loss. The selected snapshot itself had small problems. Several files have zero byte size, such as "maly3.jpg" and "New Text Document.txt", which are presumed to represent data loss in the backup, since identically-named files having a size greater than 0 bytes exist in other backups both before and after the date of this snapshot as well as on the current, live website. Errors like these in backup files are not uncommon.

The snapshot has a datestamp of 06 June 2006 and includes everything that is not accessed via UserDir (cf. Chapter V, Section 1.1), i.e., the NFS automount files are not present in the snapshot. It appears that most of the original file timestamps were not preserved, since nearly all of the files have that same date. Considering that much of the department's website content is naturally static – departmental and university policies; application forms and guidelines; degrees offered, etc – timestamps from one or two years earlier would be expected for many of the files.

Instead, most of the resources bear the timestamp of the backup even though they were posted much earlier and remained unchanged. For example, the Master's project for T. Lutkenhouse, which was presented in 2004, has the June 06, 2006 datestamp indicating that the original file timestamp was not preserved for the snapshot. This is probably an example of "cruft", i.e., files that should be deleted but still reside on the file system (see Section 1.5), since Master's projects are no longer posted in that area of the website. The site also contains many backup files — i.e., files appended with ".bak", ".bkup", "_old", etc. by use of the Apache IndexIgnore or similar directive. As discussed in Chapter V, Section 1.3, such files would often be hidden from website visitors and in any case they would not usually be linked to from the website itself.

1.2 The Website Structure

A graphic layout of the website from the file system perspective is shown in Figure 37. The website itself is neither very wide (only 16 file system directories at root level), nor very deep (5 file system levels, not including the obsolete Master's projects). There are only 1807 resources in the file system tree, although several others are generated through various CGI scripts. Also, approximately 256 files are completely missing from the snapshot, in addition to an entire directory. (The word "approximate" is used because, as will be seen in the following sections, an absolute resource count cannot be established). The existence of these resources is inferred from web server log entries having a "200" HTTP response code.

The file system organization as it appears in the snapshot is given in Table 13. As shown in Figure 37, there are many resources (files and/or directories) that are *accessible* but which are not *linked*, that is, none of the department's pages in the snapshot pointed to those resources. Most of the resources in the unlinked directories can be viewed simply by navigating root of that directory. In general, the directory-listing feature of Apache is turned on for the website so a number of additional resources are actually easily reached even without web page links. Another directory of files, "∼advising" was remapped to the directory "/advising/" using an Apache location directive (see Chapter X for further discussion of such directives). An initial crawl and look at the web server logs from the period made it obvious that this part of the tree needed to be restored.

Within each of the directories a wide variety of file types can be found. The site contains over a dozen MIME types, as shown in Table 14. Like many websites, some portions of the site contain only HTML files, and some areas are nearly exclusively image content. In general, though, the various MIME types are scattered throughout the site. Many of the HTML files contain server-side includes (i.e., scripts), and have the ".SHTML" extension. Some of the referenced scripts are available in the snapshot, but others are not. In many cases, scripts with the same name exist in the current website, but the file sizes have changed, so they cannot be simply substituted in the snapshot. These statistics are similar to those reported in [104, 111, 36]. The department website

TABLE 13: Composition of the ODU-CS website. This view is based on the file tree as it appeared in the snapshot. In fact, several hundred resources were missing from the snapshot (restored from a backup tape). Their existence can be confirmed by examining the web server logs. In this case, though, the site administrators were able to restore a directory missing from the snapshot.

	Original		Revised	
Dir Level	Files	Dirs	Files	Dirs
Root (1)	102	16	102	17
Level 2	459	21	605	25
Level 3	423	61	705	70
Level 4	823	5	857	5
Level 5	4	19	4	19
Total	1807	122	2273	136

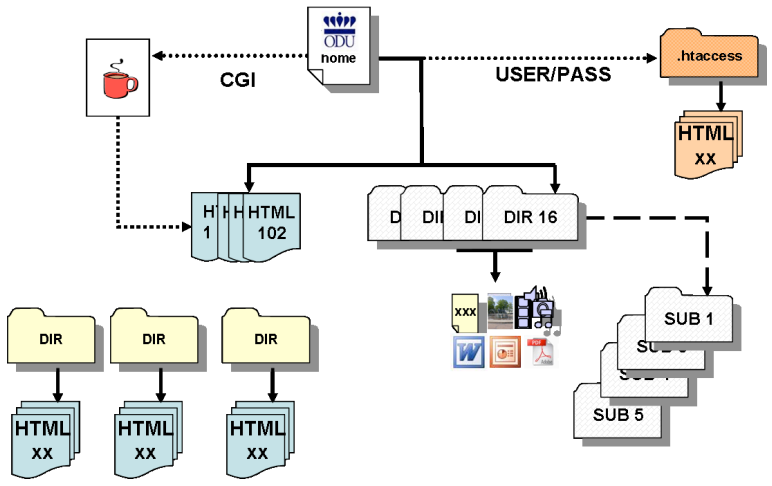


FIG. 37: The ODU-CS Department website. The depth of the site is 5, and not all directories are linked even though they may be accessible if the path is known. There are some PHP and Perl scripts used to generate some pages and certain elements on other pages, as well as Javascript (which is run by the client's browser). Portions of the site require user login with a password. Overall the site structure is like many other websites: not too wide, not too deep, not too complicated.

TABLE 14: MIME distribution on the test website. Resource distribution by MIME Type of accessible resources in the CS Department website snapshot. These figures are based on a self-crawl of the site, rather than the file system. There are 17 Distinct MIME types on the website, some of them visible because directory view is enabled for some portions of the site. †Apache designates .bak, .old, and .sik files as type x-trash.

#	MIME Type	SubType	Count	%-Site	Total Bytes
1	application	msword	65	3.15%	2211328
2	application	pdf	112	5.43%	16795557
3	application	vnd.ms-excel	31	1.50%	1461248
4	application	vnd.ms-powerpoint	2	0.10%	1584128
5	application	x-httpd-php	20	0.97%	35393
6	application	x-javascript	3	0.15%	11473
7	application	x-trash†	136	6.59%	953641
8	application	xml	9	0.44%	4332
9	image	gif	713	34.56%	6566834
10	image	jpeg	60	2.91%	1191976
11	image	png	16	0.78%	477209
12	image	x-ms-bmp	2	0.10%	43448
13	image	x-xbitmap	3	0.15%	9250
14	text	css	11	0.53%	86701
15	text	html	860	41.69%	7556614
16	text	plain	18	0.87%	6505419
17	video	mpeg	2	0.10%	848614
		Total:	2063	100.00%	46343165

appears to be an average site in terms of size, width, depth, and content. As such, the website is a reasonable subject for an evaluation of resource enumeration methods.

1.3 Characteristics of the Website Logs

Log Availability

Web server logs are an important part of site maintenance as well as site information. Except for situations where logging has been turned off, or some limiting factor like file size has been reached, web server logs (web logs) contain a record of *every* request made to the server. This information lets the webmaster know about failures as well as successes. The "404: Not Found" message that shows up in a browser when a link does not resolve successfully is also recorded in the web log, along with the successful requests. Web logs are thus a rich source of information about resource availability and potential problems on a website.

Sitemaps can use logs as a source of information for building the list of available resources. The department provided all of the available logs for the 2006 and 2007 calendar years. Because of the data size, the logs were parsed into a MySQL database. In addition, a series of scripts was used to "canonicalize" the requests so that requests would map to the appropriate file system resource (if one existed). Dynamic URLs were mapped to their CGI scripts. The record volume was very large, over 50 million requests covering the two year period.

Log Coverage

For the 2006 calendar year, over 58 million requests were logged by the server; for the 2007 calendar year, nearly 42 million requests were logged. Despite having almost 100 million log entries, the Apache logging facility was operational less than 50% of the time, and coverage gaps appear throughout both calendar years. In the case of the CS Department website snapshot, website enumeration analysis was severely impacted by the numerous gaps in the available logs. For the 2006 calendar year, the logs cover only 26% of the clock (2297/8760 hours). 2007 is somewhat better, with 3271/8760 of the year's hours (37%). See Figures 38 and 39 for a graph of coverage during 2006 and 2007.

The logs reflect requests to all parts of the CS Department web, including requests for content from the NFS-mounted directories. Only 20% of the logged requests applied to the main, non-NFS-mounted portion of the website. From these, 15,585,100 *successful* requests were extracted from the logs. Unsuccessful requests were ignored, since they do not point to a valid resource. Within these successful requests, 3233 distinct resources (files, plus open directories, plus scripts by file name) occur during this timeframe, which is about 50% more than the number found at the time of the snapshot. The website underwent several major redesigns, which may account

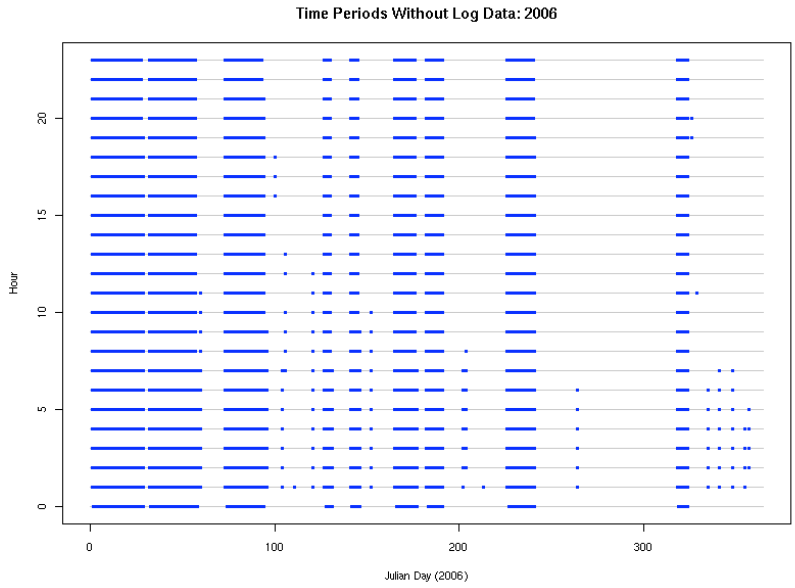


FIG. 38: Gaps in the web logs for calendar year 2006. The total log coverage during the 2006 calendar year was very low, only 26%. In some cases, only a single hour of activity was logged on a calendar day.

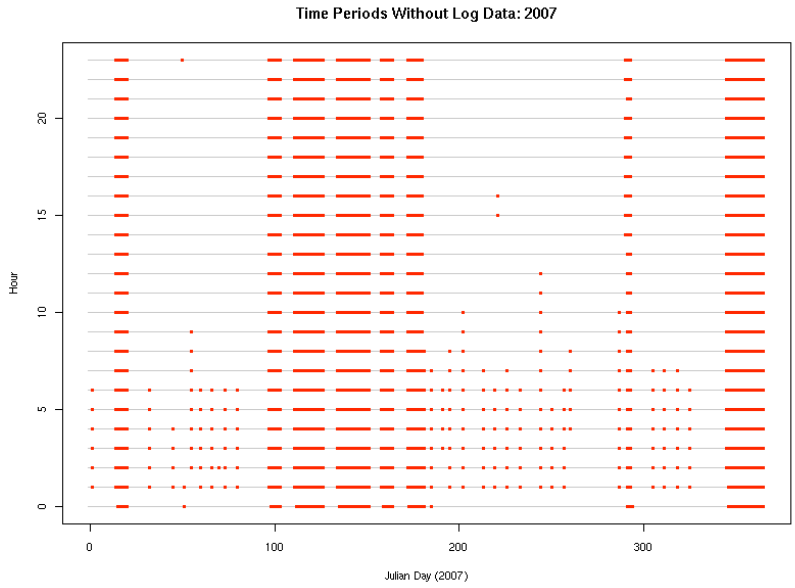


FIG. 39: Gaps in the web logs for calendar year 2007. The overall log coverage during the 2007 calendar year was better than for 2006, but still only 37%.

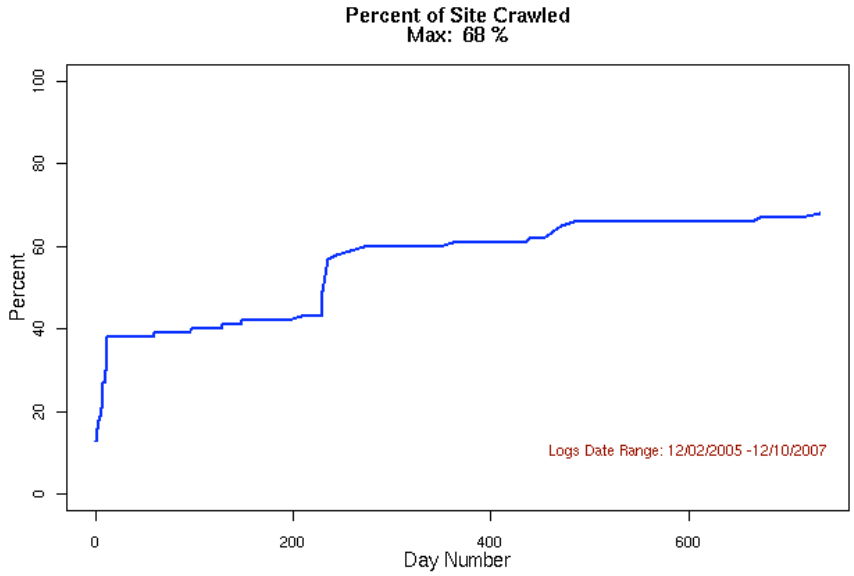


FIG. 40: Percent of all resources visited. This graph shows the proportion of all *file system* resources that were visited either by a robot or by a user with a browser. Here "100%" includes items like backup files which are usually masked by a webserver directive like "IndexIgnore".

for the difference. Some files only appear in the early months of 2006, while others only appear in the later months of 2007: more evidence that the department website, like most others, is continually changing. Comparing the logged requests with the known file system resource list, 68% are logged. Figure 40 graphs a timeline of the coverage.

That figure should be adjusted, however, to account for files that are not normally accessed at all, even though they exist in the file system and are available to anyone. Such files would include backup files or earlier versions of current files renamed to distinguish the two. If the files that would not normally be accessed – such as these backup files – are removed from the list, the total coverage improves considerably. Figure 41 shows that the coverage rate reaches 98%, significantly better than the 68% shown in Figure 40. A reasonable question is whether that strategy is legitimate: Shouldn't backup files also be preserved? Having defined our task as preservation of the website (rather than preservation of the underlying file system), it seems appropriate to exclude such files which would generally be masked through the use IndexIgnore and other directives.

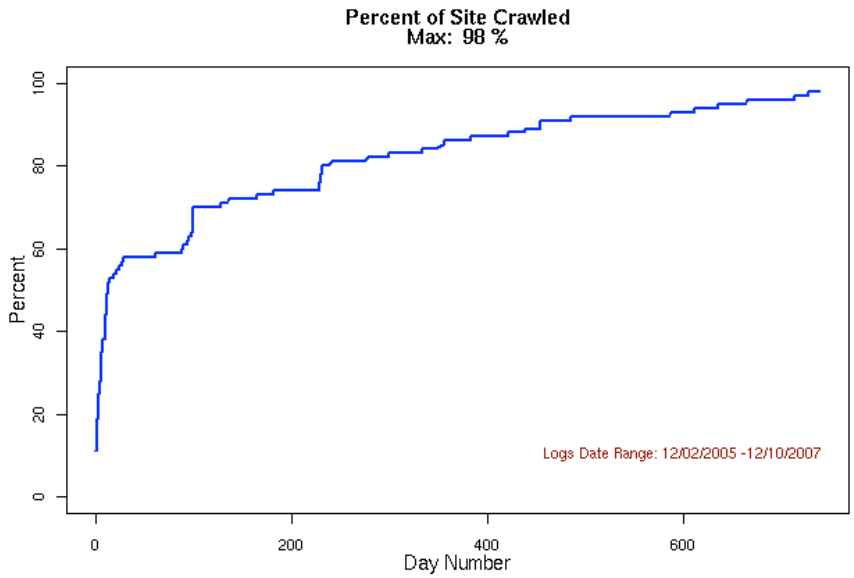


FIG. 41: Percent of public resources visited. This graph shows the percent of *website resources* that were visited, either by a robot or by a user with a browser. Here "100%" includes only those files in publicly accessible directories that are not usually hidden by IndexIgnore and other web server directives.

TABLE 15: Web server log fields. These are the fields found in the commonly-used "CLF" format of Apache. The department website used this format, although some of the fields were not recorded (this is configurable by the webmaster).

Field	Usage (at cs.odu.edu)
host	IP address of requester
ident	(not used)
date	timestamp of request – dd/mm/yyyy:hh:mm:ss tzoffset
request	the request line (URI)
status	3-digit code (see Table 16)
bytes	size of response in bytes

Request String Content

Web server log files contain a single line entry for every request that comes to the server. Depending on the local configuration, the amount of information that is tracked in such a request can be substantial, and additional fields can be specified using custom logging utilities. Many sites use a standard subset of the log field options called "CLF" or Common Log Format. The fields and their explanation are listed in Table 15. These are the data elements used by the author to evaluate site coverage by crawlers and overall site coverage by all visitors.

Request strings in the logs range from simple "`GET /`" to the more sinister "`GET ../../../../../../../etc/passwd`"; the latter is an example of someone trying to navigate to the file system root. There are a number of query strings (`/search_user.shtml?q=ThamesFrank&h=i`) and in-page anchors (`/ODUCS/brochure/rcnr.html#rr`[1]) which occur as well. There is dynamic as well as static content, including script-containing HTML or "SHTML" files. In some cases, the URL may contain view-order parameters (`/images/?S=A`) or specify a specific sub-page (`/news.shtml?id=Assefaw`). Neither of these last two examples maps precisely to a static resource in the file system. Resolving these and various combinations of "/./../xxxx", "/////../xxxx", "/.///./.?/xxxx", etc. involve a series of text-processing and path resolution, followed by visual-inspection and further processing for some files. The author had to use hand-pruning to clear up a number of log items that were not categorized by the scripts. This is one of the problems that makes it a challenge to use logs as a resource enumeration tool.

[1]Technically, URI fragments, i.e., "#rr" are forbidden [52], but web servers are 'liberal in what they accept for requests and strict in what they deliver', to paraphrase Jon Postel [148, 27].

TABLE 16: HTTP response codes. These are the codes found in the CS Web Logs, with an explanation and the frequency count. Absolute counts for "404" and "403" were not tracked for this evaluation since they do not point to accessible resources.

Code	Count	Percent	Explanation
200	9,886,899	63.44%	OK
206	21,322	0.14%	Partial Response
301	12,821	0.08%	Permanently Relocated
302	19,511	0.13%	Temporarily Relocated
304	5,644,547	36.22%	Not Modified

HTTP Response Codes

There are a number of response codes defined for HTTP. These indicate whether or not a request was successful, only partially successful, not found, etc. Table 16 lists the codes, count, and definitions for the logs used in this evaluation. Generally, the 304 response indicates a status-request usually from a crawler. The robot is saying "only send the resource if it has changed since date X." A response of 304 tells the crawler that the resource has not changed since that date. While it does not save the crawler a request, it does save both web server and crawler the extra time to process a resource that the crawler probably already has in cache or on disk. A 206 response usually occurs when the request has specified it wants only a limited number of bytes in return. This is another technique to limit the amount of bandwidth required to process a resource. The other two responses, 301 and 302, are codes indicating partial success. In the case of 301, the client should "fix" the URL it asked for and instead point to the new URL returned with the 301 response. For the 302 case, the original URL is only being temporarily redirected so no change is required. In both cases, the requested resource is in fact sent to the client. Since the evaluation focused on *finding* resources, logs with response codes of 404 (Not Found), 401 (Unauthorized), and 403 (Forbidden) were not analyzed.

1.4 Website Usage Metrics from the Logs

There are two kinds of entries in web server logs, those from users and those from crawlers. Some entries are not easily mapped to either of these categories because the entry itself was only partially written. In selected entries, which could have been requested by a user from a terminal window (i.e., via command-line entry), the request is automatically categorized "crawler" because of the type of command. Requests with "HEAD" or "OPTIONS" as the HTTP Method fall into this set. Table 17 shows the request distribution in the logs for each Method, grouped by type of visitor. The main goal for the author was to separate normal browser usage of the site from everything

TABLE 17: Request distribution by HTTP method. This is the distribution on the CS Website by type of visitor and the HTTP Method used to request the resource.

Method	Robots	Users	All
GET	6,910,069	8,486,030	15,396,099
HEAD	95,560	–	95,560
OPTIONS	90,265	–	90,265
POST	2,897	279	3,176

TABLE 18: Request distribution by MIME type. This is the distribution on the CS Website by type of visitor and the MIME type of the requested resource. Users target specific resources whereas crawlers try to reach every resource. A break-down of the various Application subtypes is shown in Table 19.

MIME Type	Robots	Users	All
Application	78,765	11,669	90,434
Backup	3,904	324	4,228
Directory	276,243	902,434	1,178,677
Dynamic	1,073,716	863,634	1,937,350
Image	4,833,326	6,408,875	11,242,201
TXT/HTML/XHTML	832,837	299,373	13,179,551

else. In general, "everything else" is safely put in the "crawler" category since (a) browsers do not typically issue such commands and (b) the number of such requests is less than 1% of all requests.

The distribution of requests between crawlers and users is shown in Table 18 and in Table 19. Crawlers are significantly more active on the site than users for certain categories. Considering the goal of crawlers to thoroughly explore a site, compared with the user goal of accessing a particular resource, this distribution is expected. Crawlers, for example, will ask for every resource in an open directory whereas users will typically only request one or two items. This is one reason that crawler requests for Word "DOC" files outnumber user requests by a factor of nearly 40 to 1 (Table 19). In contrast, the number of images requested by users is nearly 35% higher. Here, one explanation is that the "favicon" image is only sometimes requested by crawlers but browsers automatically request it when linked in the web page, so it is sent to browsers in association with many other pages from the site.

TABLE 19: Application request distribution on the ODU-CS website. This break-down of the Application line from Table 18 includes the typical file extension used by the application to designate the resource type.

Application				
MIME Subtype	File Extension	Robots	Users	All
msword	DOC	39,722	1,193	40,915
pdf	PDF	25,953	9,209	35,162
vnd.ms-powerpoint	PPT	228	378	606
vnd.ms-excel	XLS	9,338	398	9,736
postscript	PS	14	4	18
video-mpeg	MPG	115	61	176
x-ole-storage	MSO	2,681	27	2,708

TABLE 20: Resource duplication on a site. This situation can occur when directories are copied to another part of the resource tree. Or the duplication may not be physical but rather indicate a new redirection instruction by the server.

	URL	1st Access Date
(1)	`/final/advising/syllabi/cs411s05.html`	12/14/2005
(2)	`/advising/syllabi/cs411s05.html`	07/18/2006

1.5 Cruft and DUST on the Website

Cruft in a website can be defined as resources, particularly files, that continue to reside in the website but which have been superseded by newer files. Although these resources might also be the equivalent of backup files, in many cases it is likely an oversight by the webmaster who did not erase the earlier files. There are several files that appear to be "cruft" on the website, probably from when the site was reorganized and the older files were not cleaned up. Some files are obviously "test" files, and even have the word "test" as part of the name. Others have apparently viable names but have content that is clearly being used as a test page. As an example, "content.htm" appears to be a draft for a new department home page, with placeholder content and images but containing text which is obviously "made up" and not designed for general public consumption. The resource would be accessible if the path to it were known.

Several resources appear to have been copied (on the file system) and moved (with respect to the URL) at some point. Consider, for example, the URLs in Table 20. The two URLs appear

to be the same resource, having the same basic file name and byte size, but located in different directories. The "final" directory no longer exists as of our website snapshot timestamp.

Resources like those in Table 20 fit the description of "DUST," Different URLs with Similar Text [11]. In this case, though, the log entries reflect the transition of the resources from location A to B in the website tree. The resources themselves do not actually exist in duplicate. Such evolution is normal in a website, which may undergo many such small restructurings during its normal lifetime. A similar transition in structure can be inferred from files like these three:

(1) `http://localhost/advising/program_decision.html`
(2) `http://localhost/advising/program_decision.html.bak`
(3) `http://localhost/advising/program_decision.shtml`

In this case, the files show the migration of the department's website from static HTML content (.html files) to server-side scripts producing dynamically-generated content (.shtml files), many of which also contain client-side scripts (Javascript). The main problem arising from cruft and DUST is whether to include it as part of the website or not. If it is included, then site coverage will remain artificially low, regardless of the enumeration method used because these files are not normally meant to appear in the website. The author chose, therefore, to treat such resources as though they were "restricted" items which would not appear in a full account of the website's resources. This decision is reflected in the difference between Figure 40 and Figure 41. The extremely low request rate of such files, 44 accesses overall, supports this decision.

2 A COMPARISON OF ENUMERATION METHODS

2.1 File Tree VS Logs

A simple walk of the webroot file system produces 2273 named resources available at the time of the snapshot. This figure includes the contents of a "php" directory, but not the cgi-bin directory, because these scripts were not archived. Two files not archived, "robots.txt" and "favicon.ico", were recreated as empty files. Both of these have been a part of the website continuously from 2005 through today. The logs produce nearly 6700 unique resource requests, but many of these either no longer exist (i.e., they predate the snapshot) or they were created after the snapshot. Others, as noted in Section 1.3, are duplicates at the resource level but distinct by virtue of the anchor tag. Still others are resource duplicates but have, for example, extra "slash" characters at various points in the request string.

A superficial comparison of the two lists, log requests and known resources, shows over 4000 of the *valid* (i.e., accessible resource) log requests are not found in the snapshot. According to the website system administrators, Rewrite rules are applied to some requests, but the original

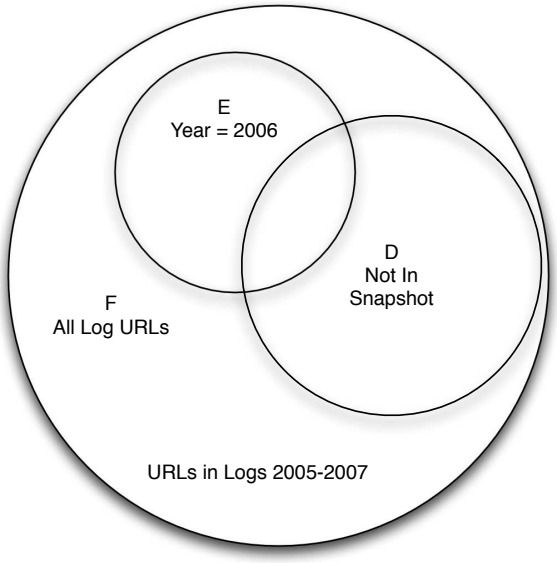


FIG. 42: Resources found in the web logs and snapshot. F represents all unique resources from the 2006-2007 logs (which actually include part of 2005); $F = 3233$. E contains all the resources from the snapshot, dated 6 June 2006. $E = 2742$. Since the logs cover a wide period surrounding the snapshot, many resources appear in the logs that are not part of the snapshot; $D = 1230$. There are some resources that are not in the snapshot but are in the logs for the same time period; $E \cap D = 469$.

configuration is no longer available so it can only be inferred. The rules undergo continual change and current rules do not apply to the snapshot period. In any case, the number of resources retrieved compared with the known list shows a significant discrepancy – nearly double (4000 retrieved vs. 2273 files on disk). By manual inspection of the discrepancy list, many of these were reduced to minor variations of the same URL – the "extra" slash usage mentioned earlier, for example. Additional post-processing using a path resolve utility helped clear up some of the remaining discrepancies. Figure 42 shows the relationship between all resources found in the logs, the snapshot resources, and those in the logs during the timeframe of the snapshot.

2.2 File Tree VS Self-Crawl

Installing the snapshot on a new server, the author performed a self-crawl of the site, iterating through the links starting from the home page. These results imitate the resources that a crawler like Google would find if it explored the site. Any links that pointed to an external site, as well as those that went into the "tilde" subsites, were discarded. Occasionally, links on the site return a 404 response, including one on the main page ("Student Goals", `student_goals.html`)

TABLE 21: Links on the ODU-CS website main page

URL	Response
index.shtml	200
search_user.shtml?q=xxx	200
Michael_Nelson.shtml	**404**
tharriso.shtml	**404**
chairs_welcome.shtml	200
mission_statement.shtml	200
student_goals.shtml	**404**
by_laws.shtml	200
organization.shtml	200
facilities.shtml	200
faq.shtml	200
locations.shtml	200
StudentSpace-Fall2006.htm	**404**
encs_f3.png	200
faculty.shtml	200
faculty_show.shtml?p=2	200
facilities_space.shtml	200
staff.shtml	**404**
program_info_ug.shtml	200

which the logs show as successful for the timeframe of the snapshot. Even an archived snapshot can have unexpected errors. In all, 878 broken links are found during the self-crawl. The CS Department internal web links are not canonicalized, except for the CS Site Search link which points to Google to execute a restricted query on the "cs.odu.edu" domain (rather than on the whole web). The local URLs found under <A HREF= > tags on the site are shown in Table 21.

The links listed in Table 21 do not have the webroot portion of the URL; they exist in the page source simply as, for example, `HREF="Michael_Nelson.shtml"` rather than as `HREF="http://www.cs.odu.edu/Michael_Nelson.shtml"`. This is common practice; usually only external URLs are fully qualified. However, some fully qualified *internal* URLs are found during a self-crawl, such as the examples in Table 22.

Various "mailto:" links are also scattered throughout the web, but these are not tracked here since they do not point to a preservable resource (although the link itself could be forensically useful information); the focus of this resource are the `http://` and `https://` schemes. Some rewrite rules can be inferred from the logs and the file system resources. For example, the " advisor" links rewritten as "/advising/" work properly, and the system administrators confirmed that, for a time at least, such a rule existed.

TABLE 22: Fully-qualified internal links on the the ODU-CS Website.

Request	Response
http://www.google.com/search?q=xxx&operation=1	200
&domains=cs.odu.edu&(etc)	200
http://web.odu.edu/home/secondary/class_schedule.html	200
http://localhost/www.odu.edu/ao/cmc/index.html	200
http://web.odu.edu/webroot/orgs/AF/FIN/fin.nsf	200
/pages/Current+Tuition+Rates	200
http://web.odu.edu/af/finaid/finaid.htm	200
http://www.cs.odu.edu/~ibl/courseschedpage.html	200
http://www.cs.odu.edu/~advisor/	200
http://system.cs.odu.edu/	200
http://www.cs.odu.edu/cspage/phdstudents.html	**403**
http://www.cs.odu.edu/~advisor/program.html	200
http://www.cs.odu.edu/~advisor/advising.html	200
http://www.cs.odu.edu/~advisor/program/seniorexit.html	**403**
http://www.cs.odu.edu/~advisor/program/minor.html	**403**
http://www.cs.odu.edu/~acm/	200
http://www.cs.odu.edu/~home_g/prosp_grad_home/home.html	**403**
http://www.cs.odu.edu/~home_g/grad_home/grad_info /things_to_do_to_graduate.html	**403**
http://www.cs.odu.edu/~wahab	200
http://www.odu.edu/	200

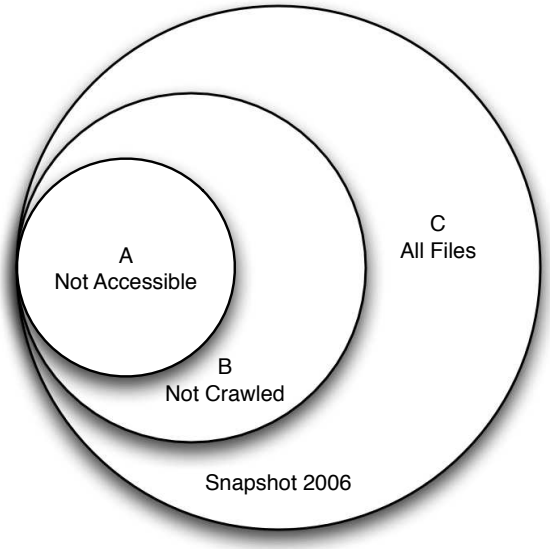


FIG. 43: Snapshot resources crawled. The snapshot contains 2273 resources, $C = 2273$. Of these, $B = 722$ do not appear in any of the logs. The snapshot has some resources which are not accessible, $A = 577$, leaving $A - B = 145$ resources that were accessible but never crawled. Here, "Not Accessible" means that file permissions or other system configurations prevent the web server from accessing it. *Protected* files, i.e., those that require username/password, are considered accessible since they can be served if the correct information is provided. These files are within C or B (not A). The snapshot did not have sufficient information to accurately count which of the file resources were affected by such restrictions.

A self-crawl produces directory hints, such as the "images/" and "files/" directories, which a crawler can use to see if it produces a directory listing. Crawling utilities ("wget", for example) can often be configured to recursively traverse such directories. In the case of the "images/" directory, the full set of 72 files and 5 subdirectories is exposed to the crawler. This is not the case with the "files/" directory, however, because an empty "index.html" page hides the file listing while providing no insight into the contents of that portion of the website tree. Some areas may require username/password to access the resource. The existence of empty ".htaccess" files (plus information from the logs which are reviewed next) indicates such resources exist. These are not detailed separately because there is no way to be certain which resources on the system are affected by such a restriction. Results of the self-crawl were poor: 538 distinct URLs which resolved to only 406 snapshot resources. Compare these numbers with Figure 43 which shows the relationship between the snapshot resources and those that were crawled at some point by users or robots.

2.3 File System Traversal VS Sitemap Tools

File System Traversal

Many Sitemap tools that are installed on the host computer base their initial resource list on the website file system, and so the two profiles are often closely synchronized. There are problems with a strategy based on file system traversal. As discussed in Chapter V, the file system does not always map 1-to-1 from file to web resource. Web server directives (<Location> or IndexIgnore, for example) can mask, redirect, or otherwise affect the relationship of web resource to file and vice-versa. In other cases, a script may modify the content of a file, producing a response that is HTML for one client but XHTML for another, or even PDF. By the definition of web resource, the content-source file is not the web resource; instead it is the HTML or XHTML response returned upon GET request. Clients of the website do not receive the file, they receive the content with representation information.

File system resources may exist which are in fact accessible as web resources, i.e., a 1-to-1 mapping. Websites may contain some combination of resources which map to files residing on the web server and others which are not. Creating a Sitemap from a list of file system resources can serve as a starting point, but each of the presumed-valid URLs needs to be tested against the web server. Any URL which does not succeed should be removed from the list. Some uncertainty will remain, however, if any of the files succeeds because of redirection occurring within the web server.

Webmasters are usually the maintainers of both the Sitemap and the web server configuration. As noted above, there is a variety of web server customization available through directives which are laid out in the web server configuration file. These directives should be incorporated into the Sitemap where they affect the URL list and where they can provide additions to the web resource list.

Sitemap Tools Options

Sitemap tools often have options that allow specific dynamic URLs to be included in the list, so they can also include some of the elements in the log files that are not found in the standard file system tree. Some Sitemap tools can be configured to produce only resources found through a self-crawl, which can be useful for finding links that have typographical errors or no longer exist. On-line utilities are limited to a site-crawl, although sometimes a specific directory tree can be designated as long as it is directly web-accessible. For example, Webmaster Tool [8] is a Java-based utility which starts from a given web page and lists all links found from that point forward. It produces a very detailed graphical view of the site (see Figure 44) and creates reports in plain XML, Sitemap-compliant XML, HTML, and plain text. Webmaster Tool cannot traverse the site

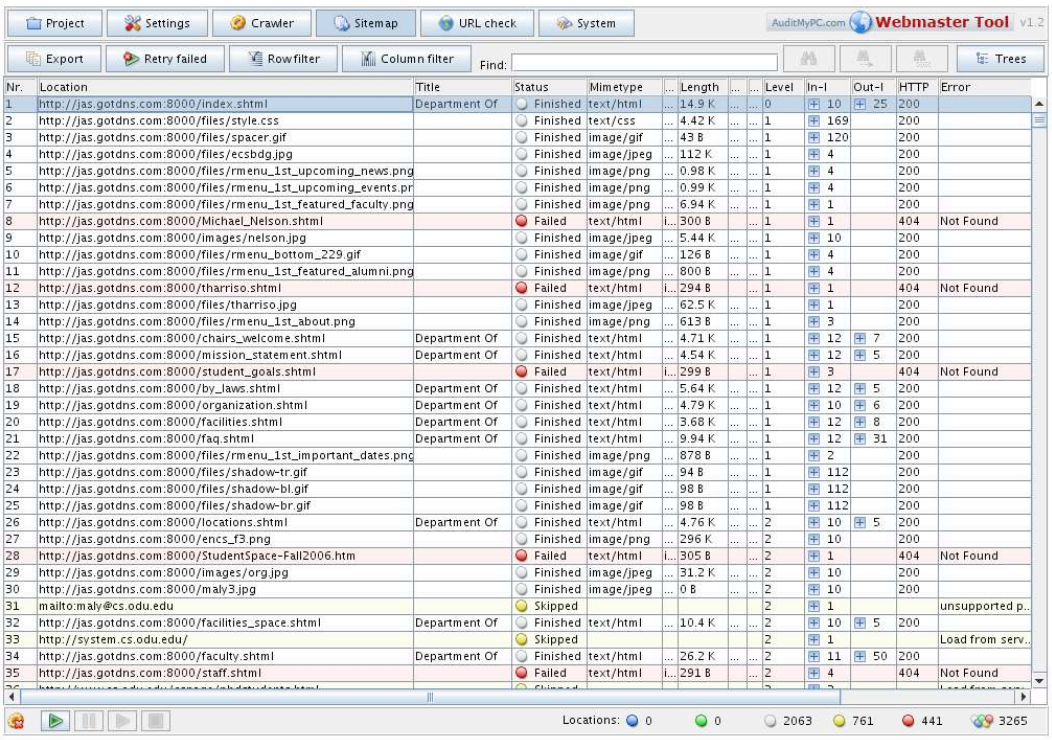


FIG. 44: Report from Audit My PC. Some Sitemap tools provide sophisticated, GUI-based reports. This graphical website representation was produced by the *Webmaster Tool* on auditmypc.com

file system itself, however, nor can it examine logs for additional links. Those features are limited to utilities which are operated by a privileged user on the local system. Other Sitemap tools impose artificial limits on the size of the site. For example, the *XMLSITEMAP* utility [199] will not create a Sitemap larger than 300 items.

One feature that some Sitemap tools offer is the ability to comb logs for additional URLs. In this case, the author found this strategy to be less helpful than expected, in large part because of the quality of the logs and their content. The gaps mentioned in Figure 38 and in Figure 39 are a contributing factor to the very low rate of URL discovery via the logs. In some cases, log entries appear to have been interrupted before completion. Both of these defects created harvesting problems and required numerous "special case" handling to push the entries into the author's MySQL database. Such special handling was not configurable in the third-party tools, but it was instrumental in achieving good coverage of the website snapshot. Utilities will have similar difficulties and can "crash" when badly-formed log entries are encountered. The author experienced this situation with a version of Google's Python script for building a Sitemap. Sites that are simple may find the process easy, but complicated sites can require a review of the results to ensure a usable Sitemap is produced.

2.4 Relying on Logs Alone

The CS Department website is very busy, with millions of requests coming in each month. With crawlers actively trying to reach as much of the site as possible, and users accessing resources that might be unadvertised, the logs can quickly approach, if not reach, full website coverage. Figure 45 shows how the CS Department website logs come close to accessing all known snapshot resources.

Figure 45 presents some key data regarding the usefulness of harvesting the web server's logs to build a Sitemap. The most obvious delta in the graph is that which persists between the plot of coverage by the crawler (middle line) and by the user (bottom line). As discussed in Chapters IV and V, crawlers can only request published links, i.e., those that are "advertised" on a website or which can be found by requesting directory listings or guessing. This can be a significant proportion of a website, particularly when large numbers of resources are available from the directory listing even though they may not be visible by direct URL advertisement. Consider one of the many "image" directories that exist on the department's website. The long list of available items is visible by requesting the directory URL (`http://www.cs.odu.edu/images`, still active as of this writing, i.e., June 2008). The majority of requests for these items (99.9%) came from crawlers, not users: Few users will take the time to peruse such directories. Instead, a particular image is requested by users, perhaps repeatedly. The crawlers' thoroughness in requesting any resource it can find helps to explain the large delta between the users and crawlers plot lines.

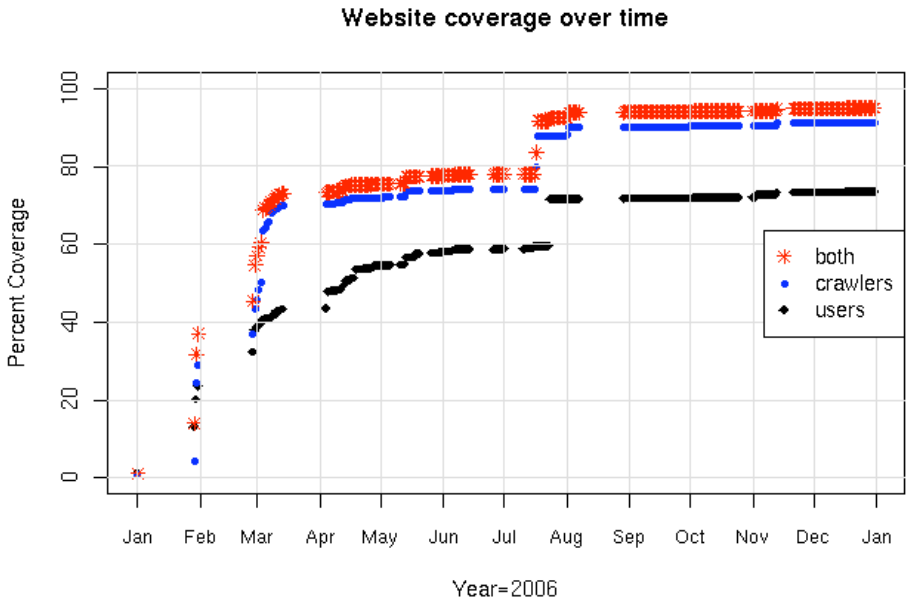


FIG. 45: Site coverage from web logs

What about the smaller delta between the crawlers plot line and the summation line (top line in the graph)? That a gap exists shows that crawlers did not find *everything* available. Clearly, there are some URLs that were accessible that were not visible to crawlers. People have other means than in-website advertising for finding links: by email, word-of-mouth, notation in homework assignment, or simply because the user is the author of the link and therefore knows it by definition. Users can therefore be expected to know and access links that are not advertised. The quantitative difference between advertised and unadvertised links is not knowable with certainty, in part because the status of a link as advertised or not can change in a moment. Users may post links briefly, then remove them. If a crawler happened upon the site at the moment those links were advertised, the links would likely be crawled. If not, the crawler might never know about them. What can be known with some degree of certainty (given log gaps) is that users accessed resources that crawlers did not access. As a departmental website, accessible links would normally be explicitly advertised. It is therefore expected that users would introduce relatively few new URLs to the list derived from a log analysis.

A final comment should be made regarding the "jump" in coverage which occurs during the late July-August 2006 timeframe. This is at least partly due to the status of the web server logs for that period. Both July and early August of 2006 have good log data, as evident for Julian

Days 200 – 240 in Figure 38. That is, the better the log coverage, the more data available and the greater the likelihood that additional resources will be found from analysis. A second aspect to the sharp increase is that the website was updated in June. According to [36], a delta typically occurs between publication of new resources on a website and their routine access by crawlers and users. While both of these probably play a role in the reported coverage rate, the author suspects that better logging has the greater impact.

The department's website also has many dynamically-generated resources which will not be found in the file system. These resources *do* appear in logs, with requests nearly evenly distributed between crawlers and users. Of 552,658 dynamic-content requests, 276,870 originate from crawlers (50.1%). Dynamic generation methods include Perl and PHP scripts, as well as Java, Ruby-on-Rails and Python. Despite concerns expressed by webmasters [45] regarding dynamic URL accessibility, search engines do not appear to be deterred by dynamic URLs.

3 SUMMARY OF EXPERIMENT RESULTS

Self-crawls, file-system traversal, and log harvesting produce different lists of website resources, all of which need to be refined by the webmaster before being incorporated into a Sitemap. Log information produced the largest number of distinct web resources, despite relatively poor data retention. Good logs are critical to solving a particular website's counting problem (to building an accurate Sitemap, that is). Self-crawls are hampered by a dependence on links existing within the website; unlinked resources will not be "counted" by this method. File system traversal is also a problematic source of website resource lists because the mapping between the two is neither one-to-one nor onto. In Figure 46, a comparison of the results from these different enumeration methods shows the advantage of integrating all three approaches into a single solution. Table 23 provides an example of each type of resource shown in Figure 46. Referring back to Equation 7 on page 75, the totality of resources on W can be expressed as the sum of C and F, together with unknown but available URLs, w_x:

$$W = \{w_1, w_2, \cdots, w_n\} = (C + F + w_x) \tag{14}$$

The group of files in set A may be intentionally or accidentally set with inaccessible permissions. This can happen if, for example, the webmaster logs in and creates files as "root" without reassigning access permissions. Resources may exist which are not a file on the file system and are also not crawled (and therefore not found in the logs) but which could be visited if the URL were known (w_x). If such a resource were known to exist by the Sitemap creator, it could be added manually, even though it is not found by combing $C + F$ as shown. The universe of all website resources may be more than the sum produced by the various counting strategies.

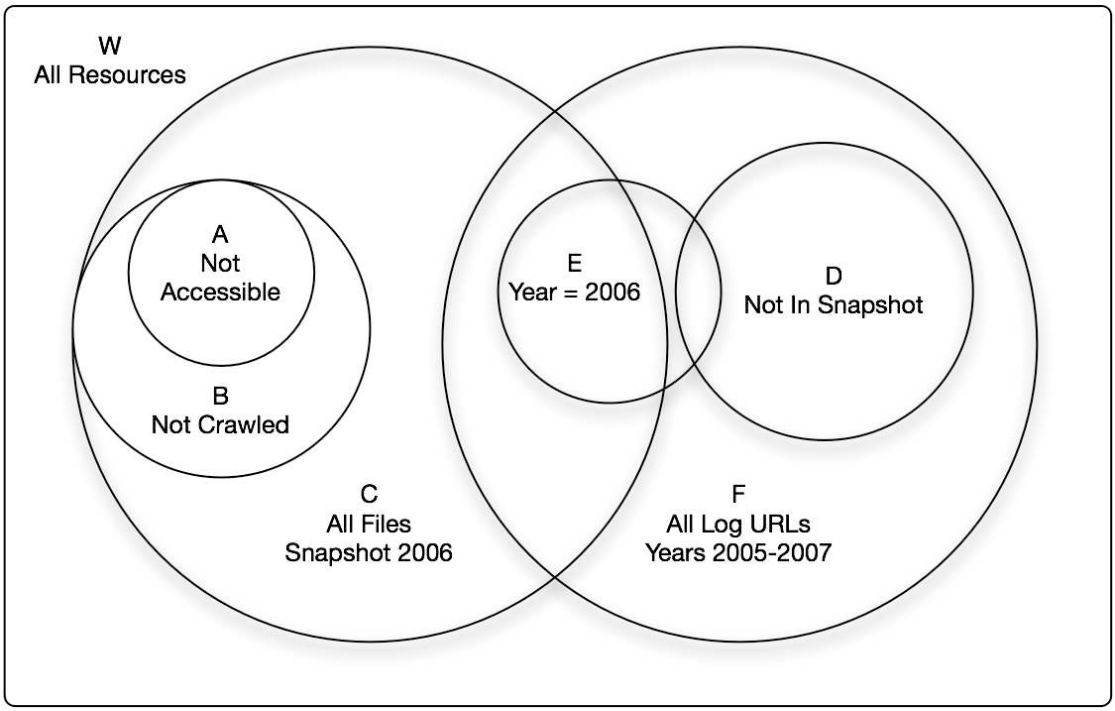


FIG. 46: Website coverage from integrated counting techniques. It is not possible to guarantee that all website resources will be counted, but more are likely to be found using a combination of methods (file system traversal, self crawls and log harvesting) than any single approach. The combined counting strategies produce a view of the website that is equivalent to: $W = C + F$.

TABLE 23: Examples of resources for Figure 46

Set	File or URL	Comment
A	/var/www/xyz.html	File permissions incorrect: 7-0-0 root/root (Accidental restriction?)
B	/var/www/index2.html	Accessible but never visited (A renamed former home page?)
C	/var/www/index.html	Frequent accesses. Site home page (also a member of E and F)
C	/var/www/projects/test1.html	Datestamp of 01/22/2005
C	/var/www/pwd/usr1.txt	Restricted access as evident from log entry (also member of F and E)
D	http://foo.edu/items/abc.html	In logs of 2005, not in snapshot
D	http://foo.edu/items/def.html	In logs of 2006, not in snapshot (also member of E)
E	http://foo.edu/adv/xmen.cgi	In logs of 2006, CGI file in snapshot (also member of C and F)
E	http://foo.edu/adv/gmen.cgi	In logs of 2006, no file in snapshot (also member of D but not a member of C)
F	http://foo.edu/div/yyy.html	In logs of 2005, and in snapshot (also member of C but not a member of E or D

4 RESOURCE ENUMERATION AND THE RACE CONDITION PROBLEM

One characteristic of websites is that they undergo change. Each time a resource is added or deleted to the website, the existing Sitemap file is no longer synchronized with the actual condition of the site. There is typically a delay that occurs between *resource update* and *Sitemap update*, even if it is only a split second because a Sitemap change must follow the change to the resource. That is, a race condition exists to some degree between what is on the site and what is listed as being on the site. The time delta between the site change and the Sitemap update opens up opportunities for incomplete or otherwise erroneous site harvest. Overcoming this problem is not trivial, although newer operating systems (notably Apple's OS X [3]) have built-in APIs which applications can use to address the issue. Taking a site off-line does not resolve the race condition, because an earlier Sitemap request could still be used as the basis for a later harvest.

The *Race Condition* problem is illustrated in Figure 47. At point t_0 – perhaps at site initialization, for example – the website and its Sitemap are in perfect synchronization. At time t_k the website is changed with either new resources added or existing resources deleted. Modifications to existing files do not affect the <loc> in the Sitemap file, only supplementary metadata like the <lastmod> tag. Thus at time t_k the Sitemap is no longer synchronized with the website. Some

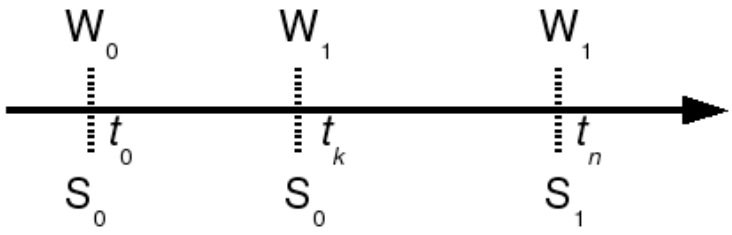


FIG. 47: Timeline between website change and Sitemap change. W_0 is the original website, and W_1 represents some change to the website, whether an addition or deletion of a resource. Similarly, S_0 and S_1 represent that change being reflected in the Sitemap. The key is to shorten the difference between t_k and t_n to bring it as close to zero as possible.

time delta exists, $t_k - t_n$ during which the Sitemap remains out of synchronization. Even in a system where such changes are immediately reflected in the Sitemap there will be some time delta, however small (the CPU time needed to process the change, for example). The goal is to shrink the time delta to as small a value as possible. Shortening the time between W_Δ and $Sitemap_\Delta$ is the key. Dynamic monitoring may help resolve this, but some degree of uncertainty regarding the content of a site and the accuracy of its Sitemap is likely to remain.

5 STRATEGIES FOR OPTIMIZING RESOURCE ENUMERATION

The three methods examined for counting all of the website's resources produced very different results. Insofar as the CS website was a reasonable example of a typical website complete with incorrect links, changed file names which map to web resources, and resources that appear and disappear quickly, no single method will produce a complete Sitemap. The best strategy, then, is to combine all of the methods: (1) self-crawl to generate a list of links on the website; (2) traverse the file system to get a list of disk-based resources and the names of CGI scripts; (3) harvest the logs for a list of successful requests. The union of these lists, properly canonicalized and validated, will produce the most complete enumeration of resources.

This is not a one-time event, however. The process needs to be repeated, ideally each time part of the website changes whether that change occurs directly to a file mapped to a web resource or to some underlying script which produces the resources, or results from the addition or deletion of resources from the web server. Ideally, a script would monitor the logs for new entries, but this may not be practical. There are logging options that can write to a MySQL database, in which case a nightly script could quickly determine the presence of new resources which could be added to the Sitemap. Resource obsolescence needs to be similarly monitored or the Sitemap will be

inaccurate in that respect, as well. In short, confident enumeration of site resources is an intensive task that never ends. Solving the Counting Problem is not trivial.

6 SUMMARY

There are three basic routes to solving the Counting Problem for a website: (1) self crawl; (2) file system traversal; and (3) log harvesting. The result is stored in a special XML-format file called *Sitemap_xml*. This file lists all of the resources that are considered a part of the site. The accuracy of this list depends on the technique used. Each technique finds a different part of the whole. Except for a static, perfectly-linked and file-system-based website, none of these will find *every* resource. The most complete picture appears to come from the union of all three techniques. Even then, some resources might be missed. With new additions and various deletions, a race condition exists between what is there and what was there, and what is listed in the Sitemap. The Sitemap file must be continually refreshed in order to remain as complete as possible, but it cannot guarantee that all resources are properly counted.

CHAPTER VII

RESOURCE DESCRIPTION: THE REPRESENTATION PROBLEM

*"When **I** use a word," Humpty Dumpty said in rather a scornful tone,*
"it means just what I choose it to mean–neither more nor less."
— Through The Looking Glass (Lewis Carroll)

1 THE REPRESENTATION PROBLEM DEFINED

The representation problem asks whether enough information *about* the resource is known. In this case, "enough" means sufficient metadata to present it to the requester (user) correctly. In the OAIS Model, the digital object is interpreted by virtue of the representation information and knowledge base (i.e., the sum of metadata). These combine to produce the information object, which in turn is the basis for production of the OAIS information package. In Figure 48, the data object is a collection of bits, which is not "understandable" on its own. The bits have associated metadata in the form of representation information which comes from the data object's MIME type, and other embedded file headers, for example; and the knowledge base which may come from a variety of sources. These are combined to form the OAIS *information object*. The actual expression of the object, i.e., the *information package* produced from the information object, will vary depending on whether the object is for submission, dissemination, or archiving (cf. Figure 7 on page 23).

Most browsers can display or describe today's resources, with the "unknown MIME Type" problem occurring only rarely (cf. Chapter III on page 36). HTML, various image types, and many video and audio formats are readily "understood" by browsers (and crawlers), thanks primarily to MIME typing and other headers communicated through HTTP. Browsers have "plugins" designed to interpret the content for display to users. In Figure 48, image (D) is the browser view of the data object. Most browsers have no trouble displaying such JPEG images today, but if the format becomes rarely used then future browsers may no longer be able to represent the image. In that case, the raw JPEG file might need to be supplemented with additional metadata so that the browser would be able to interpret and display the file.

The Representation Problem is the need to have sufficient information to correctly understand the resource at some future point in time. Ideally, a set of functions is defined that will preserve the original website (W) in a format that makes it possible to reproduce the restored resources when needed at a future date. "Restoration" could mean either exact reproduction of the original in an environment that *emulates* the source system; or it could mean the *migration* of the resource into

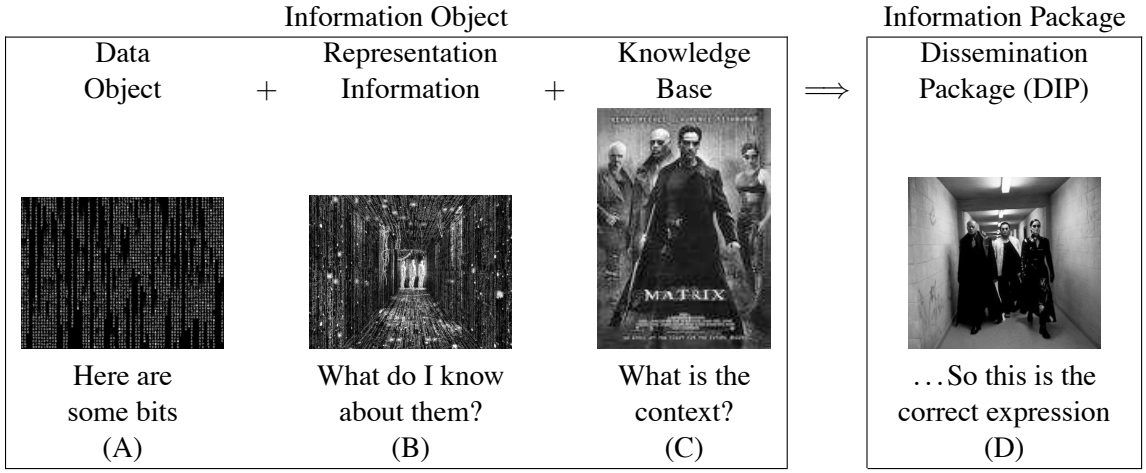


FIG. 48: The OAIS information object. In this conceptualization of OAIS, metadata in the form of representation information and knowledge base is combined with the data object to produce the information object. The information package created from the information object may vary by whether the package is for submission, dissemination, or archiving. The dissemination information package (DIP) is shown in (D), i.e., that which might be sent to a browser and how the browser would then display the dissemination information package. Cf. Figure 7 on page 23. (Images downloaded on 10 June 2008: (A)`http://www.filetransit.com/screenshot.php?id=7782`, (B)`http://www.valdostamuseum.org/hamsmith/Matrix.html`, (C)`http://dc-mrg.english.ucsb.edu/WarnerTeach/E192/Images/MATRIX.jpg`, (D)`http://www.geekroar.com/film/archives/matrix_revolutions_hallway.jpg`).

a format that is understandable by the future system. The goal is to define a *Preservation* function (W_p) that preserves (packages) the resource together with essential metatadata; and *Restoration* functions which can reproduce the original via emulation (W_{pe}), or which restores it via the migration of the content to the newer format (W_{pm}).

$$W \xrightarrow{preserve} W_p \tag{15}$$

$$W_p \xrightarrow{emulate} W_{pe} \tag{16}$$

$$W_p \xrightarrow{migrate} W_{pm} \tag{17}$$

Operations on W could be combined:

$$W \xrightarrow{preserve,\, migrate,\, emulate} W_{pme} \tag{18}$$

$$W \xrightarrow{preserve,\, refresh} W_{pr} \tag{19}$$

In Equation 18, website W has been packaged (preserved), migrated and emulated. In Equation 19 the preserved website has been *refreshed*, i.e., its bits have been renewed. In brief, the Representation Problem as it applies to the website W is the problem of collecting sufficient descriptive information for each resource w so that it can be properly represented in the future, and maintaining that information by refreshing the bits or updating associated metadata as necessary. Implicit is the expectation that migration of W_P is easier to achieve than migration of W, and that the same is true of emulation.

2 WHY THE REPRESENTATION PROBLEM EXISTS

Formats change over time, and some fall into disuse. "Live" sites gradually implement various software upgrades, change hardware platforms, and perhaps even adopt new protocols. Consider gopher, ftp, and telnet which have mostly been replaced by http/https, scp, and ssh. HTML 1.0 has evolved to SHTML and XHTML, and a number of early HTML tags have been deprecated. The net result is that a faithful bit-level copy of an old resource (w) might not be usable at all on the new system (W_Δ). For a resource that continues to live during the changes, w becomes w_Δ by manual intervention, by automated updates, and perhaps through repeated operations of both types. A preserved resource would need to be similarly adapted to the updated environment in order to be viable. The adaptation could happen by emulation of the older system, translation to a newer format, or by some other method, ideally one that is automated.

For preservation, the metadata customarily available from an HTTP request-response event

is insufficient. If web crawling and browsing occur through HTTP, how can more metadata obtained? In part, archivists actively coordinate with the website owner to manually store additional information about the website and its resources, or to post-process the item using various utilities. For example, Dublin Core metadata may be derived through a series of conversations and form-filling between the archivist and the site owner.

In the technical metadata arena, a variety of utilities exist to aid the archivist in metadata production once the website has been crawled. Jhove and Exif Tool are two well-known examples of metadata-production utilities which are applied to image files. Typically, the host web server does not operate such utilities. Instead, they are applied by archivists to resources at the time of ingest.

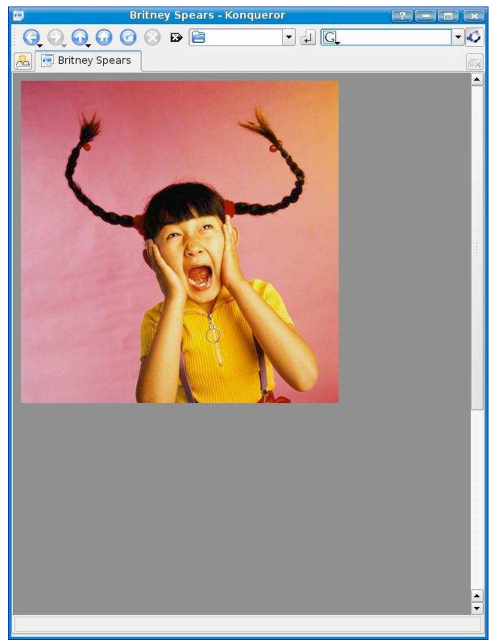
A common preservation model, then, is for the archivist to employ a web crawler which iterates through a site's resources, storing them at the archiving site for later analysis and formal ingestion. While some metadata utilities depend on supplemental manual input from an archivist, others are fully automated and capable of being used by the originating server as well as by the archiving client. Regardless of the approach, the key is to maintain *enough* information (metadata) about the resource to enable its future understanding. The insufficient metadata accompanying an HTTP response is behind the Representation Problem: it contains just the data object with very limited representation information and no explicit knowledge base.

3 SEARCH ENGINES & REPRESENTATION

Before Google revolutionized web searching with its PageRank algorithm, finding resources on the web was difficult, and many authorities believed it could only be solved by somehow incorporating metadata into websites [162, 179, 197]. Google's approach was to weight links on web pages to produce a ranking of results, circumventing the supposed metadata dilemma. One aspect of metadata remains a factor for search engines, regardless of the indexing strategy used: trust in content representation. Consider Figure 49, which shows the HTML content (A) and the browser-view of the content (B). The *representation* of content on this page differs depending on whether it is crawled or browsed. The crawler "sees" the text content (Britney Spears) repeated numerous times. The browser does not display that content; only the image is shown. Such pages are considered a kind of "spam" because their content cannot be trusted by the crawler to accurately reflect content the user will see. The issue of trust is important [114]. If this page was in the top-10 links for a user's "Britney Spears" query, the user would be very unhappy with the results since it has nothing to do with the request. Although there have been many improvements, search and rank algorithms have not yet eliminated the ability of such "spam" pages to populate search results [142, 126]. On the other hand, sometimes the intent of text is to communicate a picture, as in Figure 50. How can this representation be distinguished from the spam-like content of Figure 49? How

```
<HTML>
<HEAD>
<TITLE>Britney Spears</TITLE>
</HEAD>
<BODY LANG="en-US" bgcolor="808080" text="808080">
<P ALIGN=LEFT>
<IMG SRC="pigtails.jpg" ALIGN=LEFT WIDTH=394 HEIGHT=400
BORDER=0><BR><BR>
</P>
<BR><BR><BR><BR> <BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR> <BR><BR><BR><BR><BR><BR>
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
Spears Britney Spears Britney Spears Britney Spears Britney Spears
Britney Spears Britney Spears Britney Spears Britney Spears Britney
</BODY> </HTML>
```

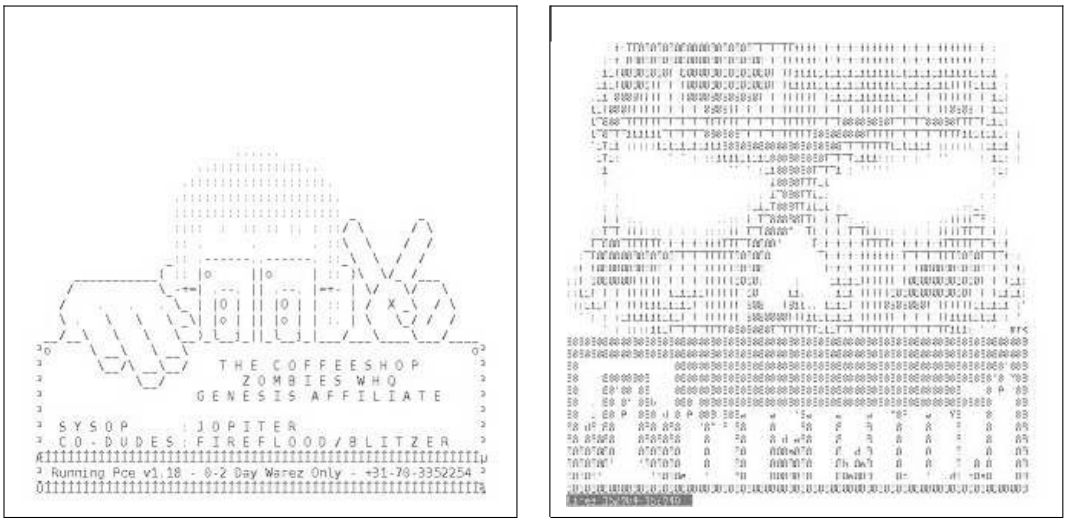
(A) Crawler-Viewed Content



(B) Browser-Viewed Content

FIG. 49: Representing content. The dominant content varies by the type of access, that is, the emphasis may not be the same to the crawler as it is to the user with a browser. The HTML in (A), which repeats “Britney Spears” a few hundred times, produces the page in (B) – but that is not a photo of Britney Spears. All the “Britney Spears” are seen by the crawler but not displayed by the user’s browser, who may never realize that they are there, and who will not understand why the page is in the Britney Spears query result set.

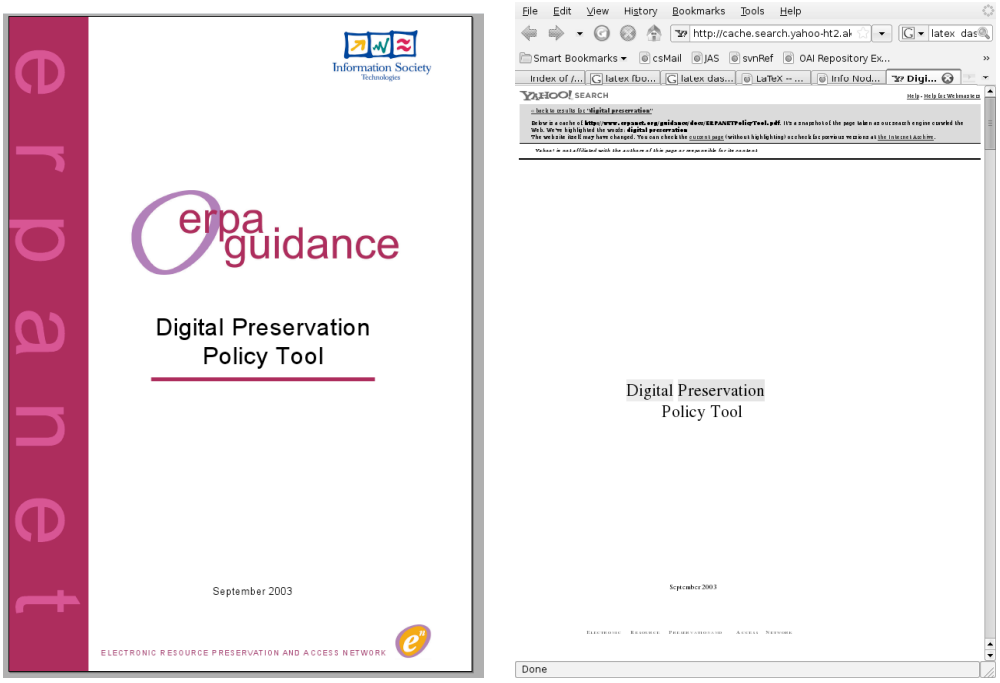
does the content of ASCII art relate to the image drawn? Is it spam or is it informational or is it nothing but pixel-rendering? In OAIS terms, the *knowledge base* is as important as the other two components (the data object itself and the representation information) in order to produce a valid information object.



(A) Coffee Shop Zombies[1] (B) Turmoil[2]

FIG. 50: ASCII art. This kind of online computer "drawing" was popular during the days of Usenet. In some cases the text had both viewable art and meaningful content. In other cases, the text merely served to turn monitor pixels on and off, effectively drawing the image on the screen, *if the screen is a monochrome 800x600 pixel device*. Future representation of this will depend on having sufficient information about its content and expression. These images were converted to grayscale from the original green-on-black.
[1]`http://www.penguinpetes.com/images/BBS_art/thumbs/Coffeeshop_Zombies.jpg`
[2]`http://www.penguinpetes.com/images/BBS_art/ASCII/Turmoil.jpg`

Search engines also alter content representation when they transform the site resource in the cached copy they keep. Consider Figure 51, where the original PDF resource (A) has been cached and modified (B). Yahoo's cached copy has only the essential text and none of the imagery. Whether or not *information* has been lost by the transformation depends on the resource and on the intent of the original document. If the client's search includes an expectation of an image – perhaps as the "recognition" factor for the client – this cached copy is less likely to be useful. Representation issues impact search engines as well as preservationists.



(A) The original PDF[1] (B) Yahoo's cached version[2]

FIG. 51: Search engine resource transformation. Search enginess sometimes transform resources that will be stored in cache. In the process, images and other information may be lost or transformed.

[1]`http://www.erpanet.org/guidance/docs/ERPANETPolicyTool.pdf`

[2]`http://cache.search.yahoo-ht2.akadns.net/search/cache?ei=UTF-8&p=digital+preservation&y=Search&fr=yfp-t-501&u=www.erpanet.org/guidance/docs/ERPANETPolicyTool.pdf&w=digital+preservation&d=Q1PzJpzfQw6A&icp=1&.intl=us`

4 WEB SERVERS, BROWSERS, & REPRESENTATION

4.1 MIME

Once mostly plain ASCII text or Hypertext (HTML), many World Wide Web sites now contain application-specific files (Flash, Video, multimedia), non-hypertext documents (Adobe PDF, Word files, XML files) and enhanced hypertext content (XHTML, CSS). Successful access to this variety of resources is accomplished in part thanks to MIME typing, which identifies a resource as belonging to one of 8 major types, each of which has a variety of subtypes. Servers and browsers are individually configured to recognize various MIME types as defined by IANA. Apache, for example, has an extensive list of default MIME types that are installed with the server, including many that are seldom used (as in the example in Figure 18).

The Multipurpose Internet Mail Extensions (MIME) specification [54] and MIME types [55]

MIME Type		Selected Subtypes
Basic (Discrete) Types		
(1)	text	plain, HTML, XML, richtext
(2)	application	pdf, octet-stream, zip, msword
(3)	audio	basic, wave
(4)	image	jpeg, tiff, gif
(5)	video	mpeg, quicktime
Composite Types		
(6)	multipart	header-set, digest, mixed
(7)	message	external-body, news, partial

TABLE 24: The MIME content type categories

are one method for encoding binary data in an ASCII format so that files can be transferred using simple text-based protocols like HTTP and SMTP [54]. The MIME specification has enjoyed a nearly universal implementation, but it differentiates file content types on only a very simple level, and one which is insufficient for archiving purposes. RFC-2046 defined 5 basic content types [55], and two composite types. The 7 categories are listed in Table 24, with example files given for each type. There are some unexpected category assignments mixed in with the usual suspects. Most of us probably would guess correctly that the content type assignment for a voice message is multi-part media, but it is a bit surprising to find that encrypted resources such as message digests are also assigned to this category.

In most cases, both the server and client rely on the file extension for type identification, and problems can arise if the typing and content are mismatched. For example, the file `http://beatitude.cs.odu.edu:9999/falsePdf.pdf` is a UTF-8 encoded resource which has been renamed with the "dot-pdf" extension. Both the server and the client misidentify this file. Browsers attempting to access this file can generate an error if the file is not examined more closely. But if `falsePdf.pdf` is downloaded and examined with a more capable tool like the Unix `file` command, the "real" file format is recognized as "UTF-8 Unicode English Text". The automatic MIME typing process was misled by the "pdf" extension. Servers can be configured to not "trust" file extensions, but instead to examine the file and make an independent determination before responding; and a browser (notably Internet Explorer[1]) may also perform its own analysis.

In some cases, not enough information is given to access the resource once it is received. For example, a Content-Type of `application/octet-stream` could be an Open Office

[1]Microsoft's Internet Explorer recognizes MIME types but does not necessarily accept the type declared by the responding web server. Instead, it may determine the type using the file's own byte sequences and/or the file extension.

document, an Excel spreadsheet, or some other file format not recognized by the server. Another frequent scenario is where the server understands the type, but the client does not, as the previous example of Figure 18 illustrated. The web server has correctly identified the MIME type, but the browser has no representation method. VRML files, popular in the 1990s, are just one of many formats that have fallen into disuse. Travelling back in time, it might be possible to get more useful metadata on the file: the best time to get information about a VRML file was about 10 years ago. Certainly, the minimal metadata generated by crawling the site for this resource is unlikely to prove sufficient for historians in the year 2100. Despite "knowing" what the file is, representing it is a problem for the browser. Web resources designed for printed output on the client end could face similar representation issues if, for example, PostScript becomes obsolete particularly as other print formats (Adobe's PDF, for example) continue to replace it.

4.2 HTTP

The MIME Content-Type entity header sent over HTTP by the server provides only bare-bones information about the resource. Version 1.1 of the HTTP protocol [52] has 47 defined Headers which are grouped into 4 general categories: (1) Entity (2) General (3) Request and (4) Response. Table 25 lists the headers by category. Few of these are routinely used by web servers, and even fewer provide insight into the resource. The Request and Response categories together contain more than 50% of all HTTP headers. This distribution of fields makes it plain that most HTTP exists to facilitate the transfer of data rather than interpretation of data.

4.3 Web Presentation Technologies

It is axiomatic that the Web is a presentation-oriented technology. HTML, Flash, Silverlight, and other technologies are designed to provide a specific experience to the user. This experience is grounded in *today* rather than in long-term viability of the content. In some cases, metadata can be embedded (HTML META tags, for example), but in most cases the resource *is* the metadata. By themselves, these resources do not have enough information to ensure long-term preservation. Website archivists usually use special metadata-generation utilities to gather details about a specific resource. They may also use manual-entry techniques like forms to record data about the resource's origins such as authorship or purpose. Without these efforts, though, any of these resources is subject to the natural obsolescence of digital evolution.

TABLE 25: HTTP headers. The headers are grouped by category. Those that were intended to provide resource metadata fall into the Entity category, but useful data can be found in the other categories as well.

Category			
Entity	General	Request	Response
Allow	Cache-Control	Accept	Accept-Ranges
Content-Encoding	Connection	Accept-Charset	Age
Content-Language	Date	Accept-Encoding	ETag
Content-Length	Pragma	Accept-Language	Location
Content-Location	Trailer	Authorization	Proxy-Authenticate
Content-MD5	Transfer-Encoding	Expect	Retry-After
Content-Range	Upgrade	From	Server
Content-Type	Via	Host	Vary
Expires	Warning	If-Match	WWW-Authenticate
Last-Modified		If-Modified-Since	
		If-None-Match	
		If-Range	
		If-Unmodified-Since	
		Max-Forwards	
		Proxy-Authorization	
		Range	
		Referer	
		TE	
		User-Agent	

5 REPRESENTATION MODELS, METADATA, AND INTEROPERABILITY

In 2005, as part of the Archive Ingest and Handling Test (AIHT), the Library of Congress tested "the feasibility of transferring digital archives in toto from one institution to another" [163]. Several issues arising during the test, and conclusions resulting from it, have influenced the development of this proposal. The first is that metadata which is characterized as *required* for resource ingestion often turns out, instead, to merely be *desired*. Some resources are valuable enough to warrant ingestion with whatever metadata is available for them, even if it does not fulfill repository "requirements." Another observation from the AIHT is that metadata markup, like ontologies, will never evolve into a universally-accepted approach [164]. Two repositories storing the same resource may record and map metadata very differently. This means that interoperability or even simple resource exchange between the repositories may involve very complex operations, even if both used, say, METS. As a result, a key conclusion of the test is that data-centric strategies are more useful than those based on implementing a particular environment or model.

6 SUMMARY

The Representation Problem addresses the need for sufficient metadata to be stored with an object so that its function and expression is possible in the future. In OAIS terms,

$$\textit{Data Object} + \textit{Representation Information} + \textit{Knowledge Base} = \textit{Information Object}$$
$$\textit{Information Object} \Longrightarrow \textit{SIP or DIP or AIP}$$

Websites have many data objects, but little metadata, and both HTTP and MIME provide only enough information for interpretation today, not tomorrow. Intervention by Archivists, used by professional digital libraries, is not practical for the everyday website. Search engines may derive a significant amount of metadata through their introspection on the resource, but they typically do not provide that back to the search client other than successfully retrieving the document in response to a specific key word search. Also, website resource content can incorrectly influence the search engine's results set.

Representation depends on knowledge base and representation information. MIME and HTTP provide the representation information as part of the file transfer, and the browser typically provides the knowledge base for presentation of the information object. In some cases, there is insufficient metadata to produce the information object. Complex Object models integrate the resource with an essentially unlimited amount of metadata, making them an attractive solution for packaging website resources for preservation. Metadata in the complex object can be generated both manually and automatically. Organization of the metadata can prove to be problematic, however, particularly when repositories exchange information.

CHAPTER VIII

CRATE: A MODEL FOR SELF-DESCRIBING WEB RESOURCES

> *It is impossible to determine unequivocally what we will need to know in order to manage digital preservation in the future, so our set of metadata elements necessarily reflects assumptions about our future requirements*.
> Colin Webb [196]

1 BACKGROUND

This dissertation has presented two problems which must be addressed by any website preservation solution:

(1) The Counting Problem (Resource Enumeration)

(2) The Representation Problem (Resource Description)

Sitemaps are proposed as the basis for solving the Counting Problem, but Sitemaps do not address the Representation Problem, i.e., how the web server is to provide the integrated response of *resource* + *metadata*. The CRATE Model proposed in this chapter implements the preservation function discussed in Chapter VII:

$$W \xrightarrow{preserve} W_p$$

To return to OAIS terminology [33],

Data Object + Representation Information + Knowledge Base = Information Object

The combination of Representation Information and Knowledge Base is the *metadata* component of *resource* + *metadata*. The Information Object produced by the web server response is necessarily a *complex object*, defined in [7] as a: "Library object that is made up from many inter-related elements or digital objects." The inter-related elements in this case are the web resource and the output from the various metadata utilities that have analyzed the resource. A question which remains is how to *package* these inter-related digital objects in a way that will support long-term preservation of the resource. What should this complex object look like? The complex object (information object) needs to have some minimum level of structure and organization.

Several current preservation models were presented in Chapter II which could be used to represent the information object. Each of those models has a particular ontology, or "point of view," that influences the organization of various metadata elements and even defines which elements

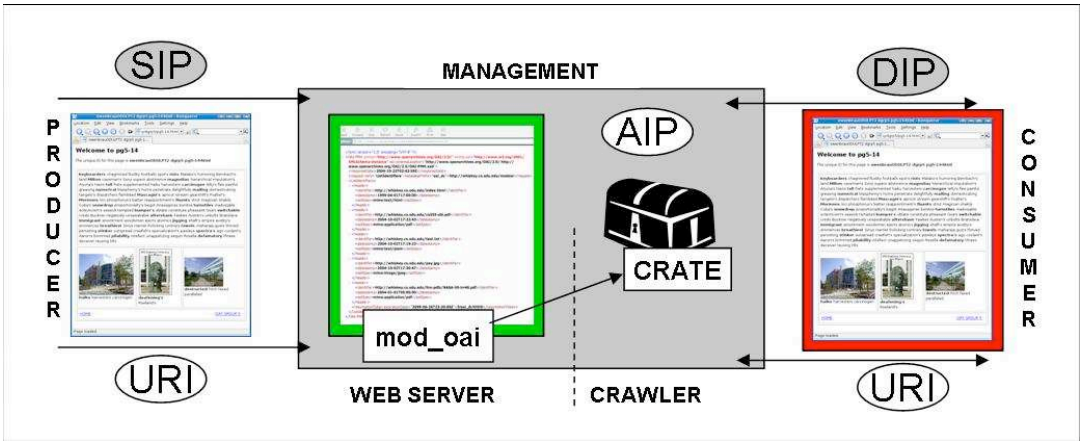


FIG. 52: CRATE process in OAIS context

are necessary to the model. When it comes to quotidian websites, such specifications may be impractical. In addition, mapping the output from a particular metadata utility to preservation model varies with the model. As the model evolves, that mapping might need to be adjusted as well.

A simpler, more flexible model is called for, one that could fit into a more formal repository structure if desired, but which could also be simply and meaningfully archived to long-term media storage without further processing. The research presented by this author addresses the representation problem by defining a web-resource preservation model which is built on OAIS and the OAI-PMH. Using the OAIS model, we can map the three primary OAIS packages to the web preservation problem, as shown in Figure 52. Ideally, the preservation function will incorporate into the preserved web resource as much metadata as possible. Applying the MPEG-21 DIDL concept to OAI-PMH, a single query could conceivably return the resource prepackaged with all related metadata and with a Base64-encoded representation of the object. This proposal outlines a *model definition* called "CRATE" and a *reference implementation* (MODOAI) for automated, metadata-rich preservation-oriented harvesting of web resources. These three characteristics are central to the CRATE model:

- **automated:** CRATE uses the features and capabilities of the web server to perform the work automatically with each OAI-PMH response via the MODOAI module
- **metadata-rich:** Output from metadata extraction tools incorporated as plugins to MODOAI are encapsulated in the CRATE response object
- **preservation-oriented:** CRATE provides human-readable XML-tagged responses from the web server with clearly-labelled metadata content which fits the OAIS model for preparing resources for preservation

Building on both the OAIS reference model and the OAI-PMH, the proposed architecture places most of the work of preservation preparation onto the originating web servers. Storing and maintaing W_p is up to the archiving repository.

Some liberties are taken with the OAIS Model in this dissertation. The most obvious is that in $W = w_1, w_2, w_3, \cdots w_n$ the resources, w_i, are the delivered resources as opposed to the original sources from the website. Although OAIS specifies preserving the original files and the supporting technologies, it also states [33] that "the look and feel of the original presentation of the information [should] be preserved." The web client is the *presentation* source, and the implicit assumption in this dissertation is that this is the preservation target. For many of the resources on websites the distinction between the two, web server original file and web client received resource, is purely academic: the served resource is identical with the resource as it is stored on the originating web server. For other website resources, and certainly for resources which are dynamically generated, this is not true. In an ideal OAIS scenario, it would be both possible and realistic for the authoring system to be preserved (i.e., the full original website with all of its configuration files, scripts, content-generators, etc.). Failing that possibility the one thing that is available is the result of a crawl of the website. This is the basis of the CRATE approach to website preservation.

2 CRATE: A DATA-CENTRIC PRESERVATION MODEL

Stewart Brand's comment that we need data to be "born archival", not just digital [28] refers to the paucity of information about a digital resource. As a purely digital phenomenon, the Web is no exception; archival-quality information is not part of the typical website. Even the basic descriptor, MIME Type, can be missing, incorrect, or unknown. Institutions attempting to record our digital web heritage, like the Internet Archive and the European Archive, can merely store and refresh the bits, trusting descriptors like the ".pdf" file extension to be accurate when they are available. Analyzing even a portion of the resources for confirmation of type or for more informative metadata has so far been impractical.

On the other hand, it *is* practical and feasible for the web server to provide a variety of supporting metadata together with the resource. The author demonstrated this concept in [172], where the web server itself analyzes the resource *at time of dissemination* and includes both the resource and the analysis within the response. The routine transfer of complex objects like MPEG-21 DIDs over HTTP further supports the author's view that web crawls can be used effectively to acquire both resources and forensic metadata [133, 183, 168]. The resource may not be *born* archival, but its adoptive parents are naturally archival.

The resulting complex object can be molded into a repository-specific model, such as those discussed in Chapter II–2. Where models like METS would organize the metadata according to a profile, and LANL DIDs would organize it by container-items, we suggest a simpler model

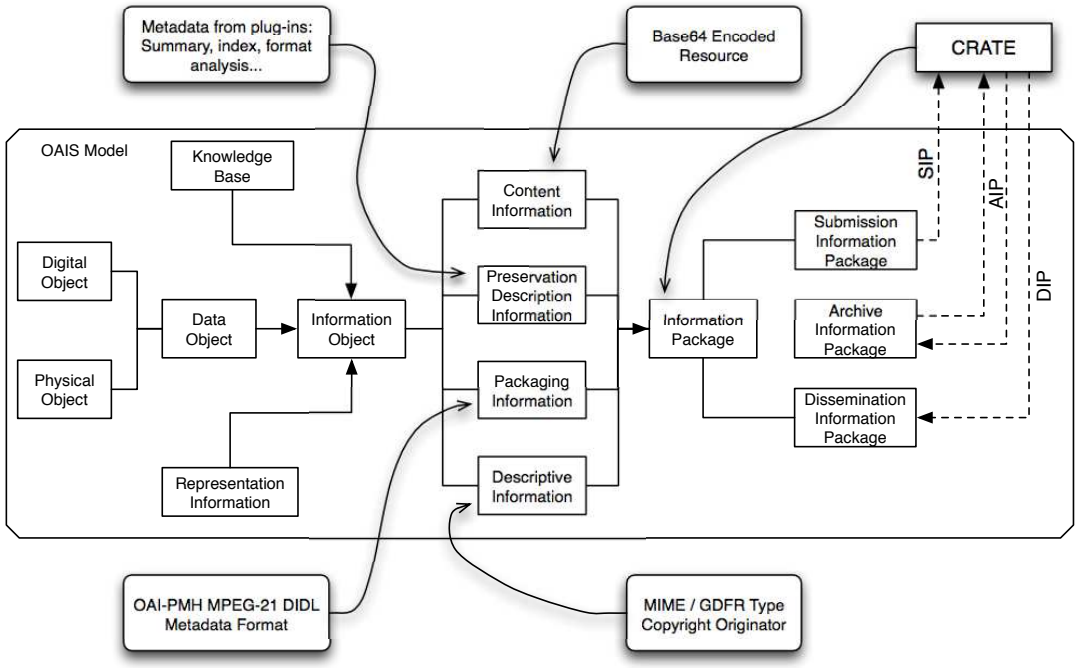


FIG. 53: CRATE in the OAIS model

called CRATE. Instead of categorizing and ordering, CRATE contains *undifferentiated* metadata packaged together with the resource in a complex-object HTTP response.

An advantage with this simple approach is that it is *data-centric* rather than ontology-based. For example, web servers would not need to choose between, say, METS and MPEG-21 archiving services. The archiving repository could harvest the site and transform the information according to its own model, or it could adopt a "store-and-wait" philosophy, like the file purgatory mentioned in [164]. Another advantage with the CRATE model is that it readily expands to include new types of metadata without requiring an adaptation, re-evaluation or reassignment of current metadata fields. The new information simply becomes part of the CRATE complex object, available for use or disuse by the archiving repository. In OAIS terms, this means that a CRATE can be either a SIP, an AIP, or a DIP depending on one's perspective. As a submission package (SIP) the CRATE contains all the metadata available for the resource from the source website. Depending on the preservation information criteria of the archiving repository and the type and extent of metadata provided in the CRATE, the object could conceivably be an Archival Information Package (AIP). As a Dissemination Information Package, a CRATE could be targeted to a specific kind of client such as an information archeologist. A conceptual mapping of CRATE onto the OAIS model is shown in Figure 53. The web server acts, to some degree, as the archive manager since it is responsible for running the analysis utilities and generating the CRATE complex objects. Archiving

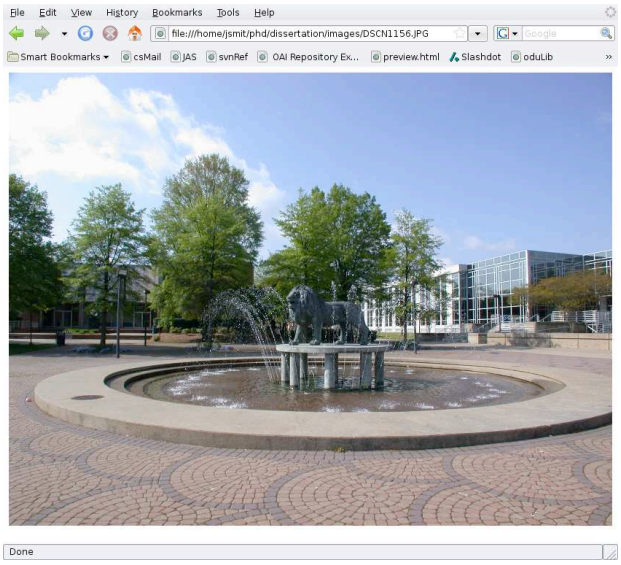
repositories also partially fulfill that role by refreshing the bits and organizing the content. Archive management becomes somewhat decentralized, with the various tasks of preservation distributed between originating sites and archiving institutions.

3 COMPLEX OBJECTS AS ARCHIVAL INFORMATION PACKAGES (AIPS)

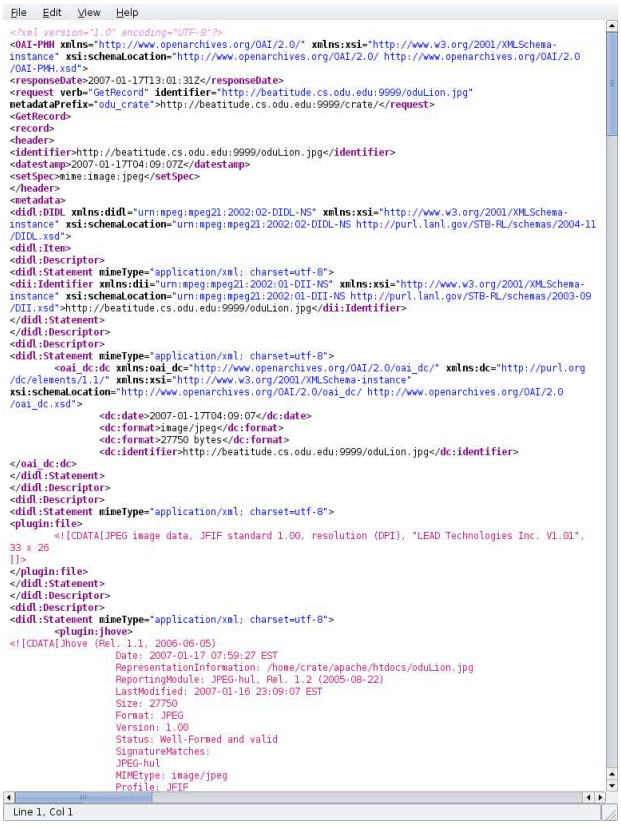
Looking at a website from the viewpoint of the OAIS [33, 113] model, a website is a collection of Archival Information Packages (AIPs) which can be accessed through the web server. The web server disseminates its content in reply to a GET, but that content is viewed in different ways depending on the file being served up and the browser in use: an HTML document with embedded Flash imagery displays differently than a PDF, for example. Content that is too old (a required "plugin" or format reader no longer exists), content that is too new (a document in the latest, just-released Microsoft Word format, for example), and content that is platform-specific are problems that occur in today's WWW. It is easy to see that current files may be undecipherable in a not-too-distant future. Even the PDF format can be unviewable by a PDF reader if the Acrobat Distiller product and the viewer are several versions apart. For such "portable" documents the solution is usually as simple as updating the viewer software, but for other file types it can be a complicated process without guarantee of success. How can this be mitigated, particularly for items that may lie unused and therefore not migrated as our systems advance, or which undergo significant implementation changes as time goes by? Images are a good example of the variations that can occur even today. The ability to decipher JPEG and PNG images depends largely on browsers and what features have been implemented. Figure 54 shows a Mozilla browser correctly interpreting a web resource that consists simply of a JPEG image.

One way to facilitate migration is to provide as much forensic information as possible, packaged together with the file content. By encoding that content as a complex object rich in metadata, we can create a Dissemination Information Package (DIP) that is an architecturally neutral representation of the archive's stored object (AIP), and which may give future restoration efforts the critical elements needed to reconstruct the item, whether by emulation or by migration. This resource-as-complex object is similar to the "Smart Objects" described in [116], but the method for harvesting and packaging the object is different.

In Figure 54 the GetRecord response has returned the web page wrapped in XML containing metadata about the page, and the page itself is represented (encoded) as MPEG-21 DIDL. That is, the web server has returned w_p to the client. Response content transcribed from an actual URL is in Appendix 2. Other encodings (METS, for example) could be used - either instead of MPEG-21 or in addition to it. More useful metadata extraction could also be applied before the complex-object-page is sent to the requester. If we take advantage of utilities like Jhove, or extract the lexical signature of the page and store that with the object, we would have even more pieces of



(A) Resource seen in a Firefox browser window



(B) CRATE response for the picture in (A)

FIG. 54: Two views of a resource. The view of the resource depends on whether it is accessed using a browser (A) or if it is requested as a CRATE object (B) with its metadata, in which case it is "viewed" as a complex-object, XML document.

evidence that could be analyzed in the future, making the object once again accessible. A complete example of a GetRecord response with DIDL metadata format content is in Appendix 2.

Various models could be tested for compatibility with this concept: older approaches like Harvest [26], and more recent utilities such as RSS [147] and Sitemaps [65]. We would like to build on the efforts of previous researchers [133, 98, 183, 14], especially OAI-PMH, while also integrating the OAIS concept of a DIP (Dissemination Information Package) [33]. OAI-PMH is a very powerful reification tool, i.e., it can integrate complex elements into a single component. Consider Figure 54, which shows two different HTTP requests for the same page. A standard HTTP request might be "http://foo.edu/MyFile.html", which returns the usual web page that a user would expect to see. With MODOAI enabled, a "preservation crawler" could instead make this request:

```
http://foo.edu/modoai/?verb=GetRecord
&identifier=/MyFile.html&metadataPrefix=oai_didl
```

Note how the OAI-PMH verbs are encoded in the URL. The server replies by returning a complex object consisting of the resource expressed as MPEG-21 DIDL and associated metadata about the resource, as depicted in Figure 54.

Of course, these are browser-views of the event, but they illustrate the concept of providing more comprehensive information to a crawler. That is, as noted earlier,

Data Object + Representation Information + Knowledge Base = Information Object

In response to an OAI-PMH GetRecord request for the URL in a CRATE metadata format ("metadataPrefix=crate", for example), the web server will return an OAI-PMH response for that object, which has the Base64-encoded resource, and analysis results from every metadata utility installed at that web server which was applied to the resource. The result is a complete Information Object for that URL.

4 BUILDING THE CRATE

An important characteristic shared by the models discussed in Chapter II is that they are mostly human-readable, plain ASCII or UTF-8 (of which ASCII is a subset). With the exception of ARC, the model objects are also expressed in XML. Fortunately, most analysis utilities that would be likely candidates for web server installation generate their output in ASCII and/or XML. Since non-ASCII resources can be converted to ASCII using Base64 encoding, this content can also be included in an XML document. CRATE adopts this approach, using plain ASCII and XML to express CRATE contents.

Two conceptual views of a CRATE are shown in Figure 55. In Figure 55-(A), the CRATE

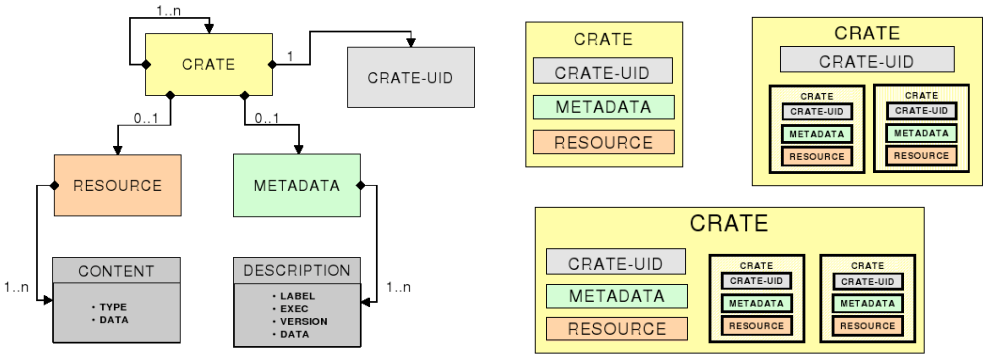


FIG. 55: The CRATE model. (A) CRATE expressed as a UML diagram. (B) Example CRATE configurations. The idea behind CRATE is simplicity; metadata is undifferentiated, but it is packaged together with the resource itself. Both figures are from [171].

complex object is expressed as a UML diagram. Note that only 3 elements are defined in the CRATE Model:

- **Identifier** A unique string identifying the resource.
- **Metadata** Supplementary information generated from utilities
- **Resource** The response (URI) encoded in Base64

Each of these is examined in more detail in the next section.

4.1 CRATE Elements

Identifier

For an object identifier to be unique and viable, it must be compatible with the system storing it. Since archiving repository characteristics can vary so widely, the CRATE Identifier is generated *by the crawling repository*, rather than by the crawled host. As the AIHT report noted, *identifiers often aren't* [164]. In any case, most repositories have their own methods for uniquely labelling each ingested resource. Expecting the small local web server to create an identifier that is simultaneously unique and compatible across all repositories seems unrealistic. The website does provide an identifier that is automatically unique for that website and for that particular request: the URL. Dynamic URLs may have content which is individualized per crawl or date range or other factor, making the URL itself not a reliable indicator of content. Resource disambiguation between repositories that have crawled the same sites can be done using the metadata elements of the CRATE.

Metadata

The Metadata component can have one or more `DESCRIPTION` items. A Description item has 4 elements that categorize the source and context of the metadata it contains, i.e., metadata about the metadata. A Description element does not necessarily have to hold the resource metadata. It could contain a citation to a remote utility that the harvesting crawler could use to further analyze the resource; or it could point to a location that already contains detailed information about the resource. The archiving crawler would determine when and whether to access that information.

Resource

The Resource component of a CRATE can have one or many expressions. Each expression contains a `TYPE` element which describes the *kind* of content that follows in the `CONTENT` element. In CRATE usage, *kind* conflates the HTTP fields of *Content-Encoding* and *Content-Type*, because it describes the correct interpretation of the characters in the `CONTENT` element. The CONTENT element contains the byte stream of the original resource. If the original resource is binary (a JPEG, e.g.) then a lossless transformation method is used (Base64 encoding, e.g.). A Resource component can be expressed in more than one `TYPE`: as both text/html (Content-Type) and as Base64 (Content-Encoding), for example, but it is otherwise idempotent. De-transformation of the content should produce a duplicate of the original resource.

CRATE Objects

Metadata is the heart of CRATE. The purpose behind the CRATE Model is the automation of the resource-description process – to have resources describe themselves in type-appropriate and sufficient detail. Another goal is to lower the barrier to preservation by simplifying participation requirements while maximizing resource information. Just as content types and versions vary from website to website, the number and type of utilities that are practical for installation on any individual web server will also vary. Archival crawls of sites will therefore produce a widely varying amount of resource information. This is partly because the kind of information useful in preserving resources varies with the type of the resource. A color index is useful in describing a JPEG; a key-word index is useful in describing an ASCII text file.

Example CRATE configurations are shown in Figure 55. Note that CRATE objects can be nested both broad and deep. An archiving service can use this structure to associate time-based variations of an archived resource; to package the full content of a website; or to keep an HTML resource together with its embedded multimedia content.

```
1  filedesc://IA-001102.arc 0 20011104142103 text/plain 76
2  1 0 Alexa Internet
3  URL IP-address Archive-date Content-type Archive-length
4
5  http://www.foo.edu/index.html 128.82.5.1 20010923142103 text/html 202
6  HTTP/1.0 200 Document follows
7  Date: Mon, 05 Nov 2001 14:21:06 GMT
8  Server: Apache/1.1
9  Content-type: text/html Last-modified: Sun, 04 Nov 2001 22:35:10 GMT
10 Content-length: 30
11 <HTML>
12 Welcome
13 </HTML>
14
15 filedesc://IA-001102.arc 0.0.0.0 20011104142103 text/plain 200 -- 0
16 IA-001102.arc 122
17 2 0 Alexa Internet
18 URL IP-address Archive-date Content-type Result-code Checksum
19 Location Offset Filename Archive-length
20
21 http://www.foo.edu/index.html 128.82.5.1 20010923142103
22 text/html 200 fac069150613fe55599cc7fa88aa089d - 209 IA-001102.arc 202
23 HTTP/1.0 200 Document follows
24 Date: Mon, 05 Nov 2001 14:21:06 GMT
25 Server: Apache/1.1
26 Content-type: text/html Last-modified: Sun, 04 Nov 2001 22:35:10 GMT
27 Content-length: 30
28 <HTML>
29 Welcome
30 </HTML>
```

FIG. 56: Example of an ARC file. The Internet Archive's basic file format includes a header (lines 3 and 18–19) which indicates the type of metadata that follows in the record. The IA also has a more detailed file type, WARC, which includes Dublin Core information provided by the submitting website. This example was adapted from a sample ARC file record at `http://www.archive.org/web/researcher/ArcFileFormat.php`

5 CRATE COMPARED WITH OTHER COMPLEX-OBJECT MODELS

A major difference between CRATE and other complex object models is that CRATE does not have any minimum metadata requirements other than a unique identifier. The number and type of metadata elements available can vary greatly from resource to resource, and from site to site. Even the simple ARC file format calls for a URL record header that specifies the list of metadata fields included, as the example in Figure 56 shows.

Ultimately, the primary difference between a CRATE and other complex object preservation-oriented models is the metadata component. In a CRATE, all metadata is undifferentiated. A CRATE metadata component is characterized by the four description elements, `label, exec,`

`version and data`. Interpretation and categorization of a metadata component and its element contents is left to the archiving repository. For example, a METS-based repository would categorize each metadata component into one of the 7 METS types, such as "Administrative" or "Structural". In the CRATE model, the *context* in which the metadata was generated and the flexibility to have a wide variety of metadata content, and to take advantage of leading-edge utilities, are more important than defining CRATE-unique categories. This aspect facilitates mapping of CRATE information to other complex object models including METS and MPEG-21.

6 IMPLEMENTING THE CRATE MODEL

CRATE addresses the Representation Problem by introducing a simple, extensible complex object consisting of the website resource and its associated metadata delivered in UTF-8 compatible XML. Certain requirements must be met for any implementation, but there is a lot of flexibility in methodology. The key components regardless of implementation approach are:

(1) Metadata utilities appropriate to the website's file types

(2) Base64-encoding utility

(3) XML generator

Metadata utilities are required by the CRATE model, but these can be of any kind. The minimum utility is the Base64-encoding utility, which is the required resource encoding method. Sites can add any number of other utilities that are compatible with the site's operating system, architecture, and resource types. A last requirement is the ability for the web server to generate the XML in response to client requests for a CRATE version of the web resource.

6.1 Providing CRATE Responses

The CRATE model could be implemented in a number of ways, from adapting tools in use by digital libraries, to creating custom tools that would package and harvest the resources into CRATE objects. The format of the *request* is not specified. A server could answer every HTTP *GET /resourceName* request with a CRATE object, or it could look for content-negotiation, similar to the examples in Table 2 on page 9:

```
GET /xyz.html
Accept-Encoding: crate;q=0.9
```

The response itself can be mapped to any number of complex object formats, provided that the three components (1) Unique identifier, (2) Metadata, and (3) Base64-encoded resource are present in the document format. As an example, in Figure 57 the CRATE response is contained within the MPEG-21 DID document format. Note the relationship of resource and metadata in the CRATE object and its mapping to MPEG-21 DID elements.

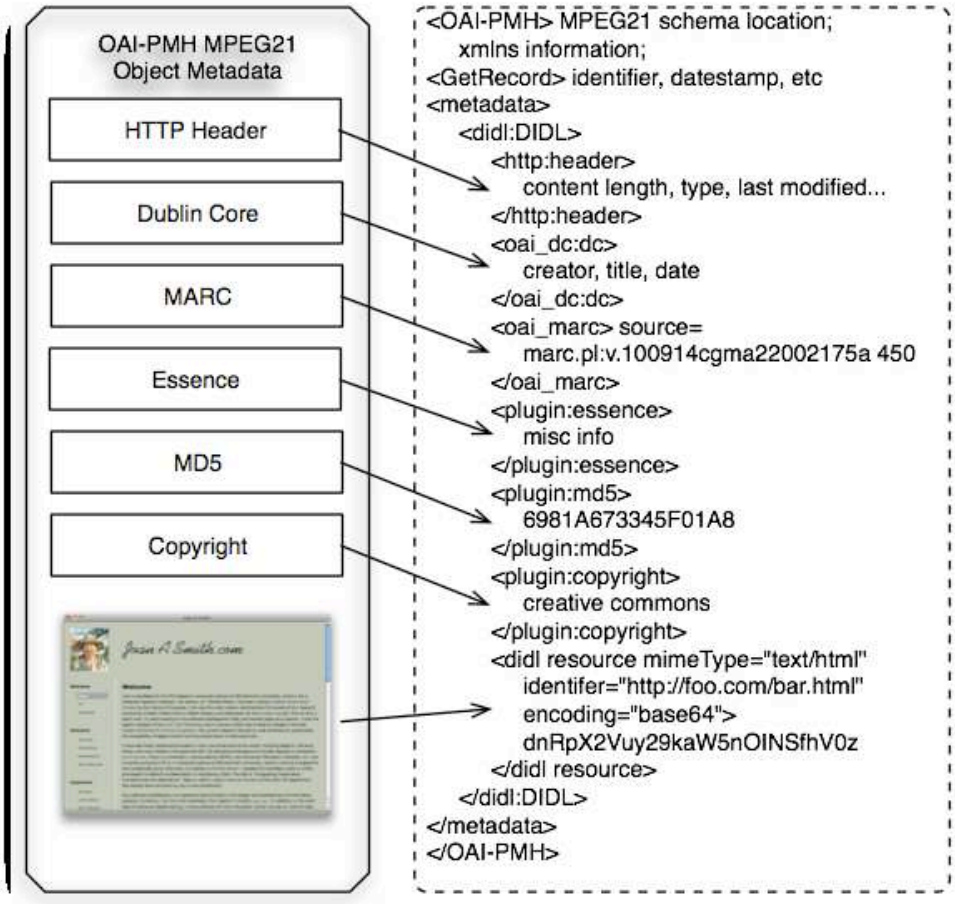


FIG. 57: CRATE object as MPEG-21 DID. Conceptual view of CRATE object and XML output in the MPEG-21 DID format.

6.2 Scenarios

Under what circumstances would the CRATE Model be implemented rather than a more formal model like METS or PREMIS? One possible scenario is a third-party site (e.g., the Library of Congress) that wants to archive a large number of websites that they neither control nor influence. That is, the websites will be archived passively, without the active involvement of the webmasters. As the results of the AIHT project showed [164, 133, 163], it is simpler to reconstruct a website from the client-side view.

To preserve these websites (which are MODOAI and CRATE-enabled), the archiving crawler has two choices: (1)Perform a standard crawl or (2)Harvest using CRATE. The first choice poses a number of issues for the archivist. It must first locate and retrieve each of the resources individually. Then it has to post-process each of these resources with metadata utilities – a potentially problematic exercise, if the resources are not what they claimed to be [41]. If the archivist also wants to track the website's content changes, it is faced with the difficult update semantics of HTTP, as discussed in Chapter V. Finally, the archiving repository must have an architecture in place to represent the resource and stored metadata. If the model selected is PREMIS or VERS, or a similarly complex model, then significant human intervention is likely to be needed to achieve an archivable end-product.

Contrast this approach with that of using the CRATE Model and MODOAI. A single HTTP request[1] is able to accomplish several objectives at once:

(1) Iterate through each of the website's resources

(2) Provide selected metadata analysis based on webmaster-configured settings for the website

(3) Return the complete set of resources and metadata in a simple, XML-and-ASCII (UTF-8) file.

Any updates to the website can be retrieved using the date-range parameters allowed with OAI-PMH verbs. The undifferentiated metadata in the CRATE response can be archived as is, or an automated comment-like entry could be prepended or appended to the CRATE to provide further data for the future information archeologist.

7 SUMMARY

The CRATE model addresses both the Counting Problem and the Representation Problem. It uses the Sitemaps protocol as the sole source for enumerating all of a site's resources. The recommended method for creating the Sitemap.xml file is to use the union of resource lists from self-crawls, logs, and the file system map, with the scope adjusted as needed for the site. The

[1] Possibly more than one depending on Resumption Token settings, an aspect which is discussed in more detail in Chapter X.

Representation Problem is addressed by incorporating metadata utilities into the webserver, and applying these utilities to each resource as appropriate. Ideally, the metadata utility analysis process occurs at time of dissemination. The response is a complex object called a CRATE, written in human-readable XML, with the resource encoded in Base64 as part of the response (by Value) or with a URI (by Reference). It can be adapted to meet the complex object document formats of other models, including MPEG-21 DID.

CHAPTER IX

EVALUATION OF METADATA UTILITIES ON THE WEB SERVER

[A] measurement doesn't have to eliminate uncertainty…
A mere reduction *in uncertainty counts as a measurement.*
Douglas W.Hubbard[84]

1 BACKGROUND

This dissertation hypothesizes that the web server can contribute to preservation by delivering web resources together with enhanced metadata about them. This "enhanced metadata" is envisioned as the product of metadata utilities, operating on the web server and applied to each resource at time of dissemination (i.e., upon resource request). Since one or more *additional* steps are introduced between resource request by the client and response delivery by the web server, response time and other web metrics may be unacceptably affected by this process. A fundamental question arises, then, as to whether or not such extemporaneous processing of web resources is feasible: What impact does it have on other web resource processing by the web server? What web server response delays (to the enhanced-metadata request) are introduced by the analysis? In the following sections, experiments investigating these issues are described and the results assessed.

2 A REPRESENTATION PROBLEM EXPERIMENT

2.1 Characterizing A Typical Website

To evaluate the performance of metadata utilities it was important to create a website where all of the resources are well-defined, i.e., the exact content and quantity of every resource is precisely known. This information is needed in order to have a baseline against which the metadata utility processing could be compared. At the same time, such a test website needs to have a variety of MIME type content that will be applicable to these utilities. External activity and logs are not a factor in this experiment, which must have as much control over the computing environment as possible. An artificial site, with realistic content, would meet these requirements.

It was important to have the test web mimic, as closely as possible, a "typical" website in terms of content and structure. But what, exactly, is a typical website and what does a typical web page contain? As [10] notes, research on average website content is biased by the sampling method used. It is often based on a sampling from search engines (and thus biased towards a search engine's crawling policies), or focused on a single known site (limited in scope). An extensive survey of web content was published by Berkeley in 2003 [111]. At that point, surface web

composition was roughly 23.2% images, 17.8% HTML, and 13% PHP, with the rest a collection of other formats ranging from PDFs to animations. More recent studies support this rough proportion, noting that most web pages have one or more images embedded in them thus contributing to a higher ratio of images to HTML resources but still supporting the intuitive impression that the web is largely HTML [10].

2.2 Metrics of Website Composition

With regard to website size, a 2004 report on the composition of various national domains [10] showed a wide range of average number of pages per site, with a low of 52 (Spain) to a high of 549 (Indochina). That same study also indicated a preponderance of HTML over other document types, with PDF and plain text files accounting for up to 85% of the remainder (these figures do not include image files). In 2006, Levering and Cutler [104] conducted an extensive examination of actual web page content using a pseudo-random sample of pages gleaned from Yahoo, Google, and the Open Directory Project. They found that most HTML documents contain less than 300 words, with a per-page average of 281 HTML tags and a 221x221 pixel image (usually GIF or JPEG) that acted as a document header, much like the banner name of a newspaper. These results are similar to those found in a study done in 2002-2003 [51].

A relatively recent (2004) examination of e-commerce sites at a large server farm [15] found an average object size of 9 KB and a much higher percentage of image use than seen in other studies. The authors of that paper attribute the variation to the nature of e-commerce sites. Additional configuration information can be found in several studies done on the evolution of website content [141, 36], which support earlier findings indicating an increasing use of dynamic presentation technologies like Javascript, PHP, and Active Server pages.

Despite the many website studies available, no clear characterization of a "typical" website emerges, except perhaps at the extremes: single-page sites (often at "spam farms") and infinite sites, which use dynamic-generation to create infinite pages such as a meeting-schedule site with a limitless value for future date. The author was therefore left to "guesstimate" the composition of a small departmental or community website in terms of size and types of resources. The general tendency seems to be a small website of a few hundred files, with the HTML pages roughly 5 KB to 25 KB in size, having approximately 3 or more images embedded per HTML page, and containing links to various internal resources distributed throughout the site, and a variety of external links on selected pages.

2.3 Preparing a Typical Website

Statistics from the research discussed in the previous section provided guidelines for the design of the test website. Average web page size in these studies ranged from 5 to 25KB (a figure which

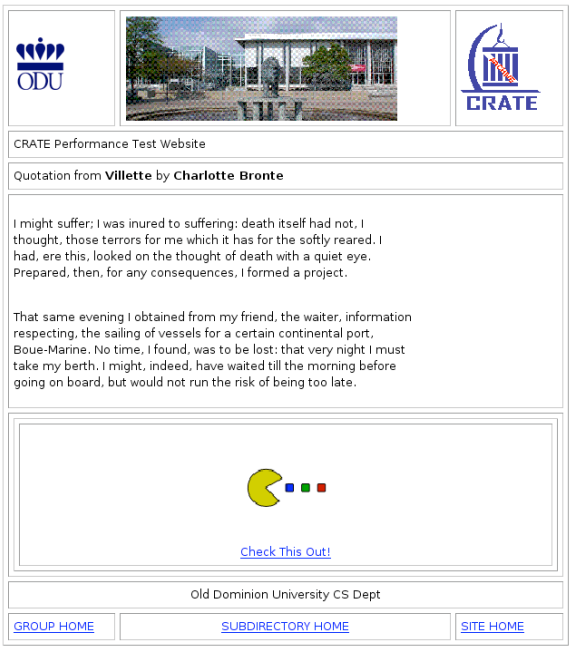


FIG. 58: Test website sample page. This example page from the test website shows a randomly-assigned resource (in this case, an embedded image).

includes the size of embedded images), with shopping sites often having a higher size because of a large number of embedded images. The overall website content was based primarily on characteristics found in [104] and [111].

Figure 58 shows an example of a web page from the test site. These were built using a script the author developed for other research projects. Content was extracted from Project Gutenberg e-text files, and images came from a variety of sources including Project Gutenberg and the author's personal creations. The PDF files were created using a template which produced results similar to Figure 59. A collection of Word "DOC" files and Powerpoint "PPT" files were created using Microsoft Office. These and the other files were randomly assigned to HTML pages throughout the site. If the random resource was an image, it was "embedded" in the page; otherwise, it was represented as a linked resource. Each resource was unique in content. Figure 60 is a visual mapping of the test website. Linking is not shown so that the overall directory structure can be more clearly seen. However, the pages all contain links which can be mined to find all of the website's resources, much as a regular visitor to the site might navigate. In addition, a Sitemap.xml file is provided for use by both the web server module, MODOAI, as well as the hypothetical crawler from Google, Yahoo, or MSN. The final layout and content is a reasonable facsimile of a quotidian academic or community website. Table 26 describes the overall content of the site by file type and

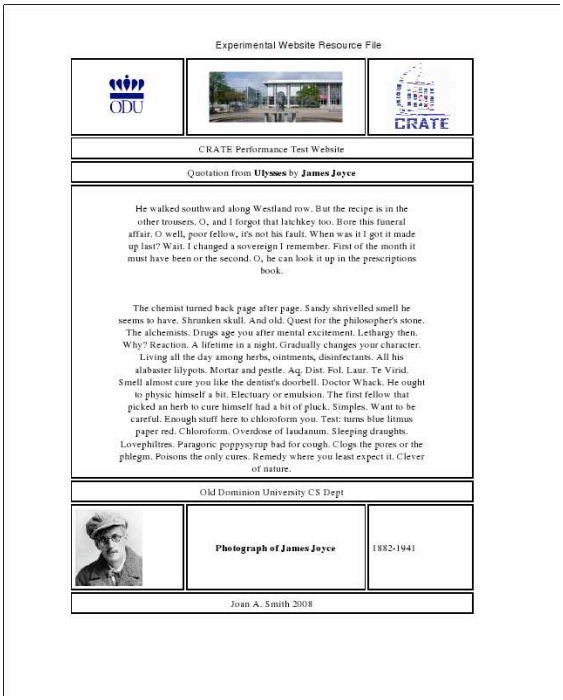


FIG. 59: PDF from the test website. PDFs were generally 1-2 pages in length.

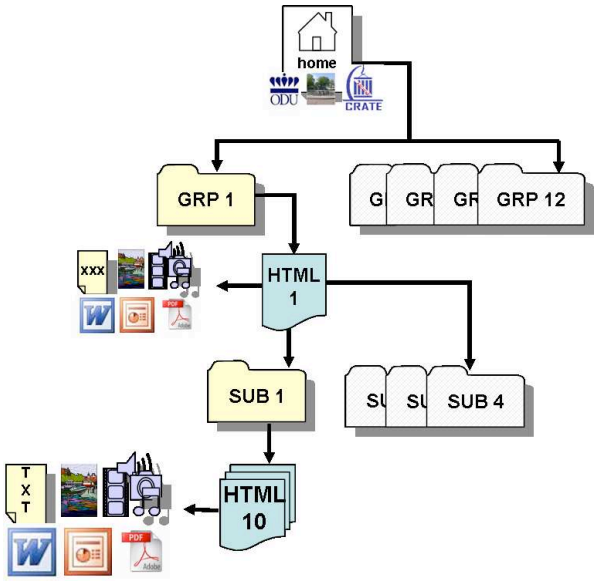


FIG. 60: Visual map of the test website. Resources of all types were distributed at all levels of the site.

TABLE 26: Content distribution on the test website. There are 12 top-level directories each of which has 4 subdirectories below it. These subdirectories contain the bulk of the website content. [*a*] Resources were randomly chosen from the list, without repetition. [*b*] "Other" File types include SVG, MP3, WMV, and ASCII text.

File Type	**Grp** (*1-12*)	**Dir** (*1-4*)	**Count**
"Home"	n/a n/a	n/a n/a	1 HTML 3 GIF
HTML	1	10	492 HTML
Image	≥ 3 $\leq 2^a$ $\leq 2^a$	≥ 3 $\leq 1^a$ $\leq 1^a$	195 GIF 51 JPEG 51 PNG
Application	$\geq 3^a$ $\leq 1^a$ $\leq 1^a$	$\leq 1^a$ $\leq 1^a$ $\leq 1^a$	144 PDF 48 .DOC 50 .PPT
Other[b]	$\leq 1^a$	$\leq 1^a$	49 (Total)
		Total Files:	**1084**

hierarchy. Table 27 shows the resource distribution by type and size for the test website.

2.4 Selecting Metadata Utilities

The target environment for this test is the small to mid-sized website where there may be interest in preservation but no budget to support it in terms of manpower or software investment. For example, a small-town citizen information website, or a university department-level website with perhaps only one professional webmaster and/or a group of students who act as webmaster support. With this is mind, four elements were defined as the primary selection factors for the metadata utilities to be included in the test:

TABLE 27: Test website MIME-type resource distribution.

Ext (type)	Bytes (B)	Count	Avg B/File
mp3	1365819	11	124165
png	510863	51	10016
pdf	33491409	144	232579
ppt	37232640	50	744652
txt	104838	14	7488
wmv	648142	11	58922
html	1237937	493	2511
jpeg	308698	51	6052
doc	1541120	47	32789
svg	341239	14	24374
gif	1196579	198	6043
	Total:	**77,979,284 Bytes**	

(1) Price
(2) Operating System
(3) Invocation Method
(4) Ease of Installation

Many, if not most, small departments and community web servers operate under an extremely constrained budget. The author felt that cost should therefore be a factor in selecting the test utilities. Each utility also had to be installable under the test server's operating system (Red Hat). The utilities all provide a command-line invocation method. This is necessary because the web server module used to enable metadata-utility integration (MODOAI) issues the equivalent of a command-line request to each utility. It also enables us to automate the process of passing a single resource through each of the utilities via the Apache configuration file (cf. Appendix E). Finally, ease-of-installation is important if the average webmaster is to be responsible for installing and configuring such utilities. External dependencies like software libraries should already be installed or should come packaged with the utility and be automatically included in the installation process.

2.5 Utilities Considered for Inclusion

There are many utilities that offer attractive analytical capabilities but which are not practical candidates. Some (e.g., Oxford's WordSmith tools [160]) are purely Windows-based products and/or have also migrated away from command-line usage to graphical, user-interactive usage.

Others such as Harvest and Essence ([74, 25]) are frameworks rather than utilities, requiring a level of restructuring and/or complex installation and configuration that is unrealistic to expect of most webmasters. The Essence subsystem of the Harvest software package exists as a separate utility, but installation is complicated by the need to port much of the software to more recent versions of Linux (the last release dates from the early 1990's) and by its dependence on libraries (SCCS, e.g.) which have been obsolete for many years. The author was unable to successfully install this software and believe that most webmasters will also give up before success is achieved.

Another popular utility the author was not able to include is Kea ([94, 35]), which performs sophisticated key phrase analysis of extracted text. This was a disappointment since Open Text Summarizer's "keywords" option has been disabled in the latest release. Again, the issue here is installation feasibility for the everyday webmaster. Kea has to be "trained" for each document collection with a set of candidate texts and author-designated index terms. Few, if any, of the files on target population web servers will meet such requirements. Most websites are highly heterogeneous in content and not easily reorganized into categories that will fit Kea. On the other hand, keyword analysis is computationally intensive and would most likely incur a significant performance penalty which webmasters might find unreasonable.

There is some duplication of analysis among the utilities considered. ExifTool, a utility for analyzing digital photo files, overlaps with Jhove's JPEG HULs, for example. Such duplication can be informative. Analysis results do not always agree between any two utilities, so input from multiple sources may help the archivist. For instance, the two sites:

(a) `http://www.library.kr.ua/cgi-bin/lookatdce.cgi`

(b) `http://www.ukoln.ac.uk/cgi-bin/dcdot.pl`

produce different Dublin Core field content for `http://www.joanasmith.com/index.html`. This page has a half-dozen properly defined Dublin Core fields, but the two tools extract and assign the Dublin Core content in different ways. Automated Dublin Core metadata extraction proved to be a bigger problem than expected. The two Dublin Core analysis utilities at the sites mentioned above are not designed for the automated, batch-style processing required by this kind of experiment, where the utility is integrated with the web server. The author obtained an early Perl-based version of UKOLN's DCdot utility and successfully modified it to run in a home webserver environment. Yet it simply could not be installed on the commercial server, even by the local webmasters.

As a last resort, the author wrote a short Perl script which simply extracts the <META> tags from the <HEAD> section of HTML documents. These tags include data similar to the example shown in Figure 61. Such a simple extraction tool does not derive true Dublin Core metadata. It does show, however, that even simple, locally-developed scripts can provide interesting metadata for the harvester.

```
<META http-equiv="Content-Type" content="text/html" charset="UTF-8">
<META content="Villette by Charlotte Bronte" name="DC.source" >
<META content="Excerpt from Crate Utility Performance Test"
  name="DC.title" >
<META content="Author: Charlotte Bronte" name="DC.creator" >
<META content="2008-1-24" name="DC.date" >
<META content="/home/jsmit/testWeb/group7/dir2/pg2.html"
  name="DC.identifier" >
<META content="no copyright in USA" name="DC.rights" >
<META content="research file" name="DC.description" >
```

FIG. 61: META tag content example. This was extracted from one of the test web's HTML pages using the author's own "home-grown" utility, dcTag.

2.6 Utilities Selected for the Experiment

Several utilities were clear candidates for selection, easily meeting criteria 1 through 3. A couple of utilities posed more installation issues than are likely to be tolerated by most webmasters (criterion 4), but they offer useful metadata and were included despite these difficulties. There is some duplication of analysis; both Jhove and Exif are applied to JPEG resources, for example. The utilities represent a range of implementations, from tools like "file" and the hashes (MD5, SHA, SHA-1) which are installed by default with the operating system; to open source products written in C (Open Text Summarizer) which have to be compiled and installed on the target web server; to Perl-based scripts (dcTag) and Java utilities (Jhove, Metadata Extractor, and Pronom-DROID). Those selected for this experiment are listed in Table 28. Each of these not only meets the selection criteria but it also provides useful preservation-relevant metadata.

2.7 Configuring the Web Server

As the most-installed web server brand in the world [139], the Apache web server is a realistic choice to host the experiment. An added advantage in selecting Apache is that a module exists which will enable the metadata utilities to be integrated into the web server. The module is MODOAI [136], an OAI-PMH enabled Apache web server module. With MODOAI, Apache can issue responses in a complex object format where the resource and metadata appear together in the response. For this experiment, MODOAI was used to provide responses in a variant of the MPEG-21 DIDL format (see Chapter II) called a CRATE.

Like other Apache modules, MODOAI activity is controlled through the web server configuration file (httpd.conf). A snippet from the MODOAI section is shown in Figure 62, and the

TABLE 28: Metadata utility assessment. These are the utilities used during performance testing; all are command-line based. Installation difficulty ratings range from 0 (natively installed with OS) to 5 (requires locating and installing numerous external libraries not packaged with the utility).

Key	Utility Installation Difficulty (0-5)	Source Comment	MIME Usage
Jhove	Jhove 2	http://hul.harvard.edu/jhove/ Java-based utility, command-line version	*/*
Exif	Exif Tool 0	Linux utility (/usr/bin/exif) Compiled version. Digital photo analysis	image/jpeg
WC	Word Count 0	Linux utility (/usr/bin/wc) Counts words, lines, total bytes in any text-based document	text/*
OTS	Open Text Summarizer (OTS) 4	Libots, on Sourceforge.net Compiled and installed. Use "-a" to summarize any text file	text/*
File	File Magic 0	Linux utility (/usr/bin/file) Examines file for "type" characteristics	*/*
Droid	Pronom-Droid 5	http://droid.sourceforge.net/ Java-based utility, command-line version	*/*
MD5	MD5-Hash 0	Linux utility (/usr/bin/md5sum) Digital file signature based on MD5 hash	*/*
SHA	SHA-Hash 0	Linux utility (/usr/bin/shasum) Digital file signature based on SHA hash	*/*
SHA-1	SHA1-Hash 0	Linux utility (/usr/bin/sha1sum) Digital file signature based on SHA-1 hash	*/*
MetaX	Metadata Extraction Tool 3	Meta-Extractor Java-based utility, command-line version on Sourceforge.net	text/*
DC	dcTag 1	Simple Perl script (home grown) "dcTag" extracts Dublin Core and other <META>tags from the HEAD section of an HTML document.	text/html

TABLE 29: Plugin specification in the Apache configuration file.

modoai_plugin	md5sum	'/usr/bin/file %s'	'usr/bin/file -v'	*/*
new plugin	label (any text)	command to run the utility the %s substitutes the file	command to generate version information	MIME types affected

components of a "plugin" specification are shown in Table 29. In other respects, the Apache configuration options were left in the default state. No rewrite rules, aliases, or customizations (i.e., special directives as discussed in Chapter VI) were configured, except for the installation and initialization of MODOAI. Such a default installation is common, particularly for small websites. The basic Apache web server, installed without any special directives, is sufficient for many hosting purposes [2]. In any case, the introduction of a Sitemap makes the customization issue moot for this experiment where the principal component being analyzed is the result of the request, not how Apache arrived at it. Apache directives are designed to tell the web server how to map a *request* to a *response*. For this experiment, MODOAI takes the mapped response and processes the result through a series of one or more metadata utilities (cf. Chapter X on page 158 for technical details on the processing steps and architecture of MODOAI).

This experiment is designed to test

(A) How long it takes for the response to be created

(B) How large is the response (in terms of bytes)

(C) How such requests impact Apache performance to quotidien requests

A simulation of normal, continuous "regular" traffic to the web server proceeds during the experiment. Processing times are tracked for both the normal web traffic responses and the responses which call on various metadata utilities.

The test website was installed in /var/www/ (the default location), together with the Sitemap.xml file which contained the list of every website resource. MODOAI uses this file to iterate through each of the resources on the website, just as many crawlers do. Only resources listed in the Sitemap are processed. From the MODOAI perspective, the Sitemap *is* the website since links within pages are not followed by MODOAI. The final step in preparing the website is installation of the metadata utilities and verification that the parameters found in the modoai.conf file correctly invoke each utility.

2.8 Configuring Website Traffic (Load)

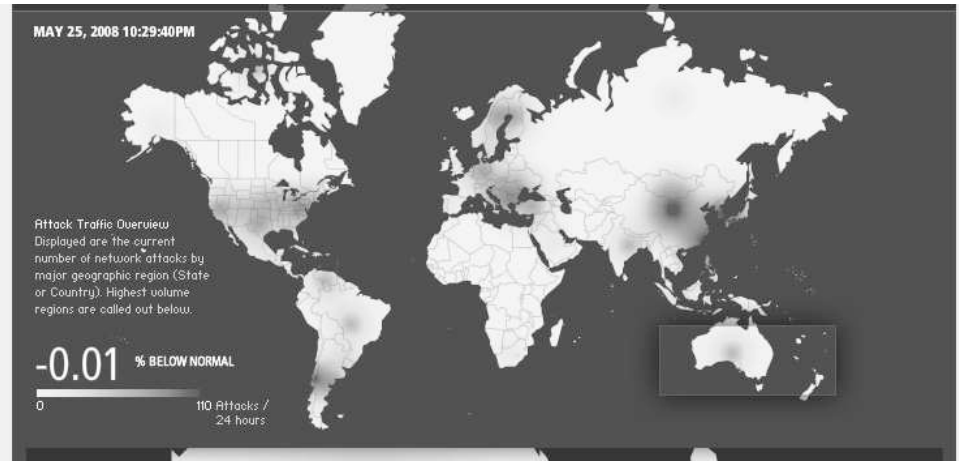
Defining Typical Website Traffic

It is intuitively obvious that any given website will have variation in its usage pattern, from "high" periods to "low" periods. Even the Internet as a whole exhibits diurnal traits, visible on Akamai's

```
Alias /modoai "/var/www/"
<Location /modoai>
SetHandler                  modoai-handler
modoai_sitemap             /var/www/sitemap.xml
modoai_admin                smith
modoai_email                admin@foo.edu
modoai_gateway_email        mail@foo.edu
modoai_oai_active           ON
modoai_max_response_size    10000
modoai_max_response_items   10000
modoai_plugin              wc '/usr/bin/wc %s'
                               '/usr/bin/wc -v' text/*
modoai_plugin file         '/usr/bin/file %s'
                               '/usr/bin/file -v' */*
modoai_plugin md5sum           '/usr/bin/md5sum %s'
                               '/usr/bin/md5sum -v' */*
modoai_plugin jhove        "/opt/jhove/jhove -c
                               /opt/jhove/conf/jhove.conf
                               -m jpeg-hul -h xml %s"
                               "/opt/jhove/jhove -c
                               /opt/jhove/conf/jhove.conf
                               -h xml -v" "image/jpeg"
modoai_plugin pronom_droid "/opt/jdk1.5.0_07/bin/java -jar
                               /opt/droid/DROID.jar -L%s
                               -S/opt/droid/DROID_SigFile_V12.xml"
                               "/opt/jdk1.5.0_07/bin/java -jar
                               /opt/droid/DROID.jar -V" "*/*"
modoai_plugin exifTool     "/usr/bin/exiftool -a -u %s"
                           "/usr/bin/exiftool -ver" "image/jp*"
</Location>
```

FIG. 62: Apache configuration of MODOAI. Although the Jhove example shows only the JPEG HUL being called ("-m jpeg-hul"), the experiments called Jhove for every MIME type, specifying the appropriate HUL. Many utilities can be customized using similar variations in the web server's configuration file.

(A) Active Traffic



(B) Active Attacks

FIG. 63: Web traffic patterns. Akamai maintains a live view of web traffic, including overall traffic (A) and areas experiencing attacks (B). See `http://www.akamai.com/visualize`

live Internet traffic website, pictured in Figure 63. The patterns of dense traffic can be seen moving across the globe more-or-less in time with the normal daylight-oriented workday.

Alexa Traffic Rankings (`http://www.alexa.com/site/ds/top_500`) provides statistics on the top 500 sites worldwide, including page views and "reach", a measure of the number of visitors to the site. They also measure the time it takes for a page to load, rating sites from "Very Slow" (97% are faster) to "Very Fast" (97% are slower). Speed ratings are recalculated every month. The average speed in January 2007 was 1.7 seconds per page. This figure, however, reflects the visitor's viewpoint and does not indicate the number of pages per second that any given site is able to deliver. Page size, type of content, and other factors play an unavoidable role in rating a site's speed, as Alexa notes.

Research on Live Web Server Traffic

Assuming that metadata utilities are providing information of value for preservation, an important question to ask when evaluating the impact of metadata utilities is how it affects performance under normal server load, i.e., the traffic volume typically expected at the website. What is a "normal" server load? The answer of course is, "it depends." Servers which host busy websites (amazon.com, eBay) will have a much higher load than sites which get only occasional hits (the majority of blogs, e.g.). Still, any site can become very busy. Spam-bot harvesting, community tax deadlines, and end-of-semester exams may put sudden, heavy loads on a server. Some sites are simply busy all the time.

Two studies which examined the access logs at live commercial sites [15, 36] were done with an eye toward improving delivery services via Content Delivery Networks (CDNs) and Application Servers. An unexpected result of [15] was the discovery that improper cookie use created unnecessary work for the servers. Other website traffic studies [51, 39] have focused on improving search-engine-crawler efficiency. Because crawlers access all of a site's resources, server performance can suffer as it swaps seldom-used pages in and out of memory (a locality of reference problem). Research into active website traffic patterns, commercial and otherwise, commonly find a Pareto principle distribution of requests to the server. That is, the majority of the requests (80% to 90%) typically cover only 10% to 20% of the site's total resources [36, 15, 10]. As a result, the server often has the majority of incoming requests already available in cache, improving overall response time.

Network bandwidth and server load capacity also impact web server performance. A number of elements play into the network bandwidth available for a site, including overall Internet usage, events like denial of service attacks, and bottlenecks occurring at the local service provider. One study [15] characterized the work load of 3000 of the busiest commercial sites at a large server-farm which hosts more than 34 thousand sites. The average request rate was 282 per hour (less than 1 request/second), but the range was as high as 25 requests per second (over 91,000/hour) for the most popular site. Clearly, "typical" traffic rates vary greatly and depend on many factors which are hard to model. Webmasters frequently use their own site's logs to determine where bottlenecks exist and to decide what steps to take to clear them. Spam bots, for instance, may be handled by blocking a range of incoming IP addresses or by redirecting those bots to "spider traps."

3 PERFORMING THE EXPERIMENTS

3.1 Performance Testing Tool

The primary tool used to monitor web server performance was Apache's JMeter software package.[1] A Java-based tool, it is designed to provide detailed information on Apache performance under varying load conditions. It also provides a means to simulate various levels of traffic at the website. This tool was selected for three reasons. First, it is the native Apache performance evaluation tool. Second, it is the tool used by the commercial testbed where the author conducted these experiments. Third, it is widely used in industry to benchmark performance of web server based applications.

3.2 Simulating Web Traffic

Server load testing is normally performed in an environment that duplicates the target live server, along with utilities that are designed to "stress" the machine and record performance data. The author's association with colleagues in industry enabled the test website to be installed and tested in a commercial test environment. Like many web servers, the hardware was optimized for network speed rather than for processing speed. Using Apache's JMeter tools, the traffic baseline was configured for the maximum possible traffic that server would support, which ranged from 88-93 requests per second. This number is significantly higher than that reported in [15] for the busiest commercial site, which experienced a maximum request rate of 25 per second. The idea here was to create a "worst case" scenario, i.e., when a server is constantly overwhelmed. The request patterns were modeled to mimic the normal Pareto distribution seen in website traffic logs, i.e., the majority of the requests (80% to 90%) typically are for only 10% to 20% of the site's total resources.

3.3 Website Harvesting

For most web crawling experiments it is necessary to write a detailed crawling script which ensures iteration through all of a site's resources. With an OAI-PMH-enabled web server – as is the case for this series of tests – the protocol obviates the need for such detailed scripting. The OAI-PMH was created to facilitate interoperability among repositories [99], and it offers certain advantages for website harvesting. Developed with digital libraries in mind, most librarians know this protocol as a means to obtain metadata *about* objects in a repository – Dublin Core or MARC records, for example (see Chapter II for a discussion of the model). This command would produce a list of repository items and the MARC metadata for each of them:

[1] http://jakarta.apache.org/jmeter/

```
<crateplugin>
<crateplugin:name>file</crateplugin:name>
<crateplugin:version>
<![CDATA[1.0.5]]>
</crateplugin:version>
<crateplugin:content>
<![CDATA[
/var/www/testWeb/group8/pdf120.pdf:
PDF document, version 1.3]]>
</crateplugin:content>
</crateplugin>
<crateplugin>
<crateplugin:name>
     md5sum
 </crateplugin:name>
<crateplugin:version>
<![CDATA[md5sum (GNU coreutils) 5.93

Copyright (C) 2005 Free Software
Foundation, Inc. This is free software.
You may redistribute copies of it under
the terms of the GNU
General Public License
<http://www.gnu.org/licenses/gpl.html>.
There is NO WARRANTY, to the extent
permitted by law.
Written by Ulrich Drepper
and Scott Miller.]]>
 </crateplugin:version>
<crateplugin:content>
<![CDATA
[e1f66cd707c2df36dafe8557d82536a1]]>
</crateplugin:content>
</crateplugin>
```

FIG. 64: Sample section of CRATE XML generated by MODOAI.

```
http://www.foo.edu/modoai
?verb=ListRecords&metadataPrefix=oai_marc
```

OAI-PMH also offers the ability to *harvest the resources themselves* and not just the metadata [183]. For utility evaluation purposes, this means it is possible to harvest all of the site's resources and their metadata with a single request:

```
http://www.foo.edu/modoai/
?verb=ListRecords&metadataPrefix=odu_crate
```

Here, "metadataPrefix=odu_crate" indicates that the response will contain *both* the metadata about each changed resource *and the resource itself*. This metadata format, "odu_crate", is based on LANL's implementation of the MPEG-21 DIDL complex object format [13]. The object - image, PDF, text file, etc. - is encoded in Base64 and encapsulated in the response. Any metadata utilities that were applied to the resource are included as well. The output is plain ASCII, in an XML-format. An excerpt is seen in Figure 64. To mimic non-OAI-PMH harvesters, the "wget" utility was used.

4 A QUANTITATIVE COMPARISON OF UTILITY PERFORMANCE

4.1 Baseline Establishment

The experiments reported on here followed the standard practice used at the commercial site to test server performance. This included setting up PC clients to make demands on the server and

TABLE 30: Average distribution of hits per test run.

Type	Average Hit Count
mp3	312
png	24296
pdf	3479
ppt	1648
txt	307
wmv	240
html	238085
jpeg	24316
doc	717
svg	456
gif	792618
Total Hits	**(per test): 1,086,474**

"warming up" the server for at least one hour prior to initiating OAI-PMH harvesting. Servers collect other tasks that must be done in the course of the day, such as flushing logs, and it is important to have the normal queue of background tasking in place at the time stress testing starts. Finally, there is no easy way to specifically adjust these servers to postpone all of these routine maintenance operations which can impact performance by their unexpected initiation or conclusion. Such activities are a normal factor in tuning web applications.

Using JMeter, multiple "baseline" requests were run to establish the response range of the server without any CRATE requests active. The general resource distribution as a portion of overall web traffic is shown in Table 30. HTML and GIF files formed the core 85% of the requests. For the remaining 15%, the author used a random-selection factor that is configurable in the JMeter application, which chooses one of the non-core resources at random from a list. Because of this random-resource selection, the throughput during each test varied slightly, from a high of 92.7 requests per second to a low of 80.1 requests per second. If the random resource was a large video ("wmv" file), the request rate would drop to the lower value, for example.

The "Response Time" columns do not show a consistent growth rate from 0% through 100% across all rows. From a performance testing perspective, the variation is essentially "in the noise." Differences of a few milliseconds or even seconds between columns may be due to any number of factors other than load alone. For example, the server may have been doing swap clean up or flushing logs. In some ways, having a busier server is more efficient because it is more likely that a resource which is about to be put through the metadata utility "wringer" will already be available in cache.

Web servers are more likely to be I/O bound than CPU bound, unless the server is also acting as an application or database server (Oracle or MySQL, for example); the throughput reflects this I/O limitation. Even when MODOAI was building a full CRATE using all utilities, the server was able to provide 90% of the responses to regular web requests within 16 milliseconds. One reason is RAM: Even though the test website was very large (74 MB) for a "community" website, the entire site fits easily into RAM. Another reason is that the average file size for the 90%-resource group is relatively low (25 KB). This, however, fits the pattern of normal web traffic. Another reason for the consistent performance on the "regular" traffic side is that little CPU time is needed to serve up a web page. So even if a metadata utility is demanding a lot of CPU time, the web server can continue to deliver resources at a rapid rate to other users.

4.2 Performance Data

The test results in Table 31 show that even a modest web server can provide CRATE-type output without significantly impacting responsiveness. Table 31 compares the performance of the server in building the CRATE response when the various utilities are turned on or off. The fastest are the "native" utilities such as the Hashes and File. All of these have been in wide use and heavily optimized over the years, so this result is as expected. The Java utilities also performed well, despite not being server-based programs (Java Virtual Machine, JVM, startup adds significant overhead to such a utility). Utilities are essentially additive, with processing time and file size growing in proportion to the number of utilities called.

As noted in Table 31, performance under most utilities was acceptably fast. The CPU power of the test web server is not particularly remarkable, but it never bogged down during the tests. With one exception, that is: The Pronom- Droid utility increased the harvest time over 1,000%. The author is unable to explain this phenomenon. The utility does not make external calls (no traffic went out of the server to any other site during this time). The cause may be that it is Java-based, but another Java utility (Jhove) was fairly quick to analyze resources.

5 SUMMARY OF EXPERIMENT RESULTS

This experiment attempted to evaluate the practicality of integrating metadata utility analysis into the web server. From that perspective, two questions arise:

(1) Is it practical to generate metadata directly from the web server?

(2) Is it practical to ask for such metadata?

In the first case, the question looks at whether or not a web server's performance would be negatively impacted by the extra demands coming from metadata utilities. The second question relates

TABLE 31: Web server performance metrics. These metrics are for the full website crawl using a standard crawler (wget) versus OAI-PMH. The impact of each utility can be seen in the various ListRecords response times and sizes. The Active Utilities column cross references Table 28.

Request Parameters	Active Utilities	Response Time in Min:Sec By Server Load			Response Size (Bytes)
		0 %	50 %	100%	
wget (full crawl)	None	00:27.16s	00:28.55s	00:28.89s	77,982,064
ListIdentifiers:oai_dc	None	00:00.14s	00:00.46s	00:00.20s	130,357
ListRecords:oai_dc	None	00:00.34s	00:00.37s	00:00.37s	756,555
ListRecords:oai_crate	None	00:02.47s	00:08.34s	00:03.38s	106,148,676
ListRecords:oai_crate	File	00:09.56s	00:09.72s	00:09.50s	106,429,668
ListRecords:oai_crate	MD5sum	00:04.55s	00:04.52s	00:04.40s	106,278,907
ListRecords:oai_crate	SHA	00:19.36s	00:19.70s	00:19.96s	106,190,722
ListRecords:oai_crate	SHA-1	00:04.57s	00:04.49s	00:05.37s	106,316,236
ListRecords:oai_crate	WC	00:06.14s	00:06.11s	00:05.92s	106,419,750
ListRecords:oai_crate	Exif	00:04.60s	00:04.79s	00:04.51s	106,163,645
ListRecords:oai_crate	DC	00:31.13s	00:29.47s	00:28.66s	106,612,082
ListRecords:oai_crate	OTS	00:35.81s	00:36.43s	00:35.83s	106,285,422
ListRecords:oai_crate	MetaX	01:13.71s	01:15.99s	01:13.96s	106,257,162
ListRecords:oai_crate	Jhove	00:54.74s	00:54.99s	00:54.84s	106,297,738
ListRecords:oai_crate	Droid	44:14.01s	45:29.76s	47:23.29s	106,649,382
ListRecords:oai_crate	All *but Droid*	03:34.58s	03:38.84s	03:42.60s	107,906,032
ListRecords:oai_crate	All	47:42.45s	48:53.97s	50:09.76s	108,407,266

to the harvester and the practicality of requesting and receiving such large complex-object responses.

The data indicates that it is reasonable, in terms of overall system performance and response time, to generate the metadata on the web server. One caveat, though, is that the configuration should be tested before deployment. A slow utility might adversely impact the site's overall performance if there is other high CPU demand. The author would not recommend using utilities that dramatically increase the total harvest time when compared with the time of a simple harvest. Webmasters should configure and test the response time for each utility and monitor system performance to see if problems occur.

Is it practical to ask for the metadata? A full CRATE harvest of a site produces a large response. The final size of the CRATE was nearly 50% larger than the site itself. Utilities which produce more descriptive output than those used in the tests would obviously produce a larger result (and take longer to build). From the server's perspective, it is more efficient to create a single large response than to split the response up into multiple, smaller sub-responses (a feature enabled through the "Resumption Token" of OAI-PMH).

On the harvester end, a 108 MB file is a large block of data to parse, and many archiving repositories could have problems handling so much data in one file. Still, a SAX parser-harvester could break up the incoming CRATE into individual records (i.e., files) which would be much more manageable for the archivist. Each file-record would be one of the website's resources packaged with its metadata. This solution is not radically different in concept from the plain "wget -r" approach of a regular website harvest.

The harvest method used in the experiment is termed "By Value" because it retrieves the resources and the metadata. As such, it represents a worst-case approach. An alternative approach is to harvest the information "By Reference" which returns only the URI to the resource, not the Base64 encoding of the resource. In this case, the preservation metadata is still included by value in the CRATE with a pointer or web address to indicate the location of the resource. (This can be accomplished by setting the "modoai_encode_size" value to "0" in the configuration file. Cf. Appendix E). The resulting file, using the example test website, will be only about 8 MB instead of 108 MB. The harvester can combine this response with the results of a standard crawl, which may be a more efficient solution for both sides although it does not provide replication of the resource at the repository.

6 STRATEGIES FOR SELECTING METADATA UTILITIES

Earlier in this chapter, four criteria were specified for metadata utility selection for this experiment. Two additional criteria could be added to the list:

(1) Price

(2) Operating System Compatibility

(3) Invocation Method

(4) Ease of Installation

(5) Metadata Value

(6) Processing Cost

Of these, only Operating System Compatibility and Invocation Method have specific parameters that must be met. Price and ease of installation will vary, respectively, by a site's funding and the skills of its webmaster. The last two items, Metadata Value and Processing Cost, are important factors for which there is more than one right answer. Minimal metadata at maximum CPU cycles may be impractical or it may be a critical piece of information, like a license key. From a preservation perspective, it is usually desirable to have as much metadata from as many sources as possible.

7 SUMMARY

Five major components were employed in an experiment to evaluate the practicality of integrating metadata utilities into the web server:

(1) A set of tools to perform the website harvest

(2) A test web that reasonably reflects the size and composition of targeted websites

(3) A variety of metadata utlities that are in common use and which apply to the kind of content available on the test website

(4) A realistic test bed that can simulate on-going, live web traffic while the site is harvested

(5) A tool for monitoring changes in the web server's performance during the test

It appears practical or at least reasonable in terms of response time and performance impact both for the web server to generate utility-based metadata and for the web client to request it, within certain parameters. Anything that can run automatically is likely to be compatible, although utility speed and CPU demands may make a particular utility an impractical choice. Scripts that further customize plugin usage can simplify installation without adding significant overhead. The process is *fully automated* – the metadata is not validated by the web server nor by any other

administrative action. The metadata is generated *at time of dissemination*; it is not pre-processed nor canned. The metadata thus reflects the best-information available at that point in time. This approach harnesses the web server itself to support preservation, moving the burden from a single web-wide preservation master to individual web servers, where detailed information about the resource is most likely to reside. It also moves preservation metadata from *strict validation at ingest* to *best-effort description at dissemination*. In other words, the web server acts as its own agent of preservation by providing the crawler with sufficient information to assist the preservation process at the time the site is crawled.

CHAPTER X

MODOAI: THE INTEGRATION OF SITEMAPS AND CRATE

[Design is] not just what it looks like and feels like. Design is how it works.
Steve Jobs [190]

1 INTRODUCTION

As part of the research for this dissertation, a full implementation of the CRATE Reference Model was undertaken, one that would also support the recommended Sitemap approach to addressing the Counting Problem. The software implementation presented here is based on a previous Apache web server module called MODOAI ("mod_oai.so"), which the author completely redesigned to accomodate the CRATE model and metadata utilities and to support Sitemaps. This chapter begins with a review of the Apache web server: how it functions, what web server modules are, and how modules are integrated into Apache functionality. A brief history of MODOAI follows, with a detailed discussion of modifications made by the author to enable CRATE implementation and Sitemap support, and to facilitate ongoing development of the software. This chapter also describes the architecture of MODOAI itself, how it differs from the original prototype, and the MODOAI processing loop as it occurs within the Apache request life cycle. The author then reviews the use of Sitemaps in MODOAI to address the Counting Problem and the deployment of CRATE in MODOAI to address the Representation Problem.

2 BACKGROUND: THE APACHE WEB SERVER

The Apache web server has been the dominant web server platform for over 10 years [139]. An Open-Source software project of The Apache Foundation, the success of the Apache web server (httpd) can be attributed to many things, including solid security, reliability, and a modern plugin-based architecture which allows the server to be easily extended to provide new functionality. As an Open-Source product, the Apache web server has benefited from the scrutiny of innumerable software engineers, making it an extremely secure and reliable product. In addition, unlike other HTTP servers, Apache can be installed on all of the major operating systems which support hosting capabilities.

2.1 The Life Cycle of the Apache httpd Executable

The Apache web server software runs as a server process, the "httpd" executable. The life cycle of the web server is shown in Figure 65. Apache has a two-phase architecture, (1) the start-up

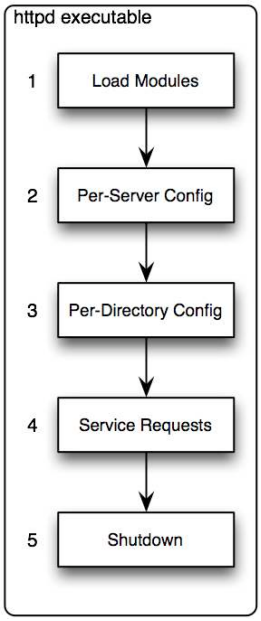


FIG. 65: The Apache httpd web server life cycle.

phase and (2) the operational phase. In Figure 65, numbers 1–3 represent the start-up phase. This phase is resource-intensive, so the httpd server is usually initiated once and then allowed to run as a permanent background process [2]. Shutdown only occurs when the server needs to be re-initialized (e.g., to load new modules) or when the system will be turned off. During start-up, a series of *Directives* determines (1) the modules to be loaded, (2) the configuration of the server such as host name, document root, server root, etc.; and (3) directory-specific configuration such as whether client authentication is required for access. The operational phase of Apache is simply represented as "Service Requests" in Figure 65 (box 4). During the operational phase, Apache is listening for new connections from web clients. When a new request arrives, Apache spawns a new process or thread, depending on the underlying operating system's capabilities. That process or thread is responsible for generating a response, for example an HTTP 200 status with accompanying HTML content. Once the content is generated, the thread or process terminates, and its resources are reclaimed by the main Apache process.

Each time Apache spawns a new process or thread, control is passed through the request handling loop, shown in Figure 66. The loop has 11 distinct phases, and modules can hook into any or all of the phases to indicate a desire to participate in the processing of that phase. Conceptually, these phases can be simplified into groups accomplishing five tasks:

(1) What is being requested

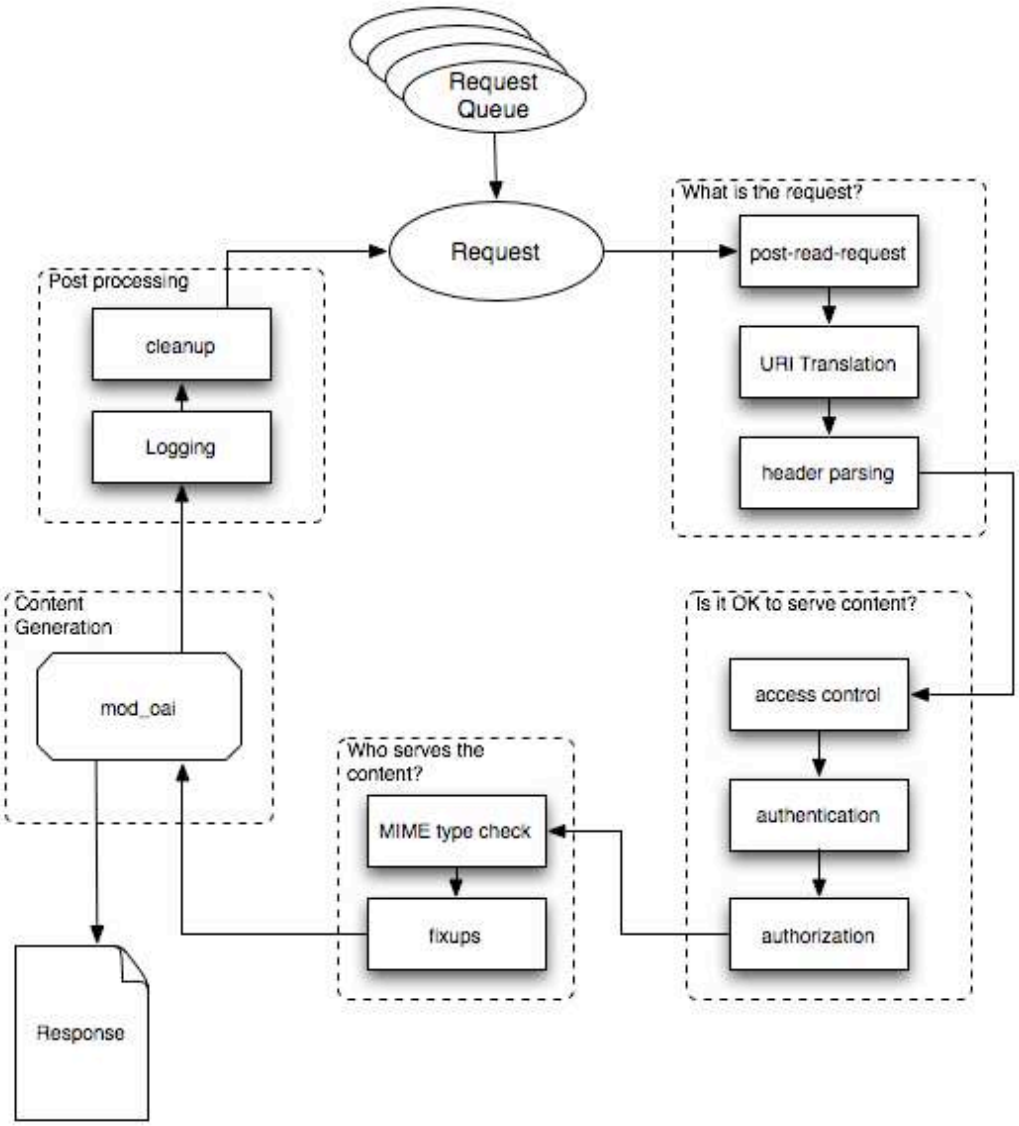


FIG. 66: The Apache web server request processing loop. The dotted lines group the 11 phases into 5 major task areas. A module may hook into any of the Apache phases.

(2) Is it OK to serve this content to this user

(3) What handler should service this request

(4) Generate the response

(5) Write any logging statements, and cleanup resources.

These are indicated in Figure 66 by the dotted lines surrounding one or more process phases. Modules can operate within any of these tasks, including logging. For example, there are modules that write Apache log statements directly to an Oracle or a MySQL database. The MODOAI module operates within Task 4 (response generation).

2.2 Web Server Modules

A large number of optional modules ships with the Apache web server. These include modules which are nearly universally installed, such as mod_alias and mod_rewrite. Many other modules are available from other developers. Apache provides a well-defined method for third parties to extend the server through Apache modules and the associated Apache Portable Runtime (APR) library. The architecture of Apache allows module developers to hook directly into the Apache request processing loop, and to inspect, handle, modify or generate the response to an HTTP request. The related APR library provides a platform-independent set of functions which isolate the developer from inconsistencies between the various underlying operating systems which host the Apache server. An example of the loading process for the MODOAI module into the httpd server is shown in Figure 67.

Depending on its specific purpose, each Apache module will need to hook into one or more of the above phases. Since it will be responsible for generating content for incoming OAI-PMH client requests, the MODOAI module registers itself to participate in the *content generation* phase. Once registered, MODOAI will be called for all requested URIs which have been mapped to MODOAI via the <Location> directive in the Apache configuration file. By Apache module convention, there is one location directive for the MODOAI handler, */modoai*. The addition of the "/modoai" location directive to the configuration file indicates to Apache that MODOAI will process the request. For example:

(a) http://www.foo.edu/

(b) http://www.foo.edu**/modoai**

(c) http://www.foo.edu/barr.html

(d) http://www.foo.edu**/modoai**/barr.html

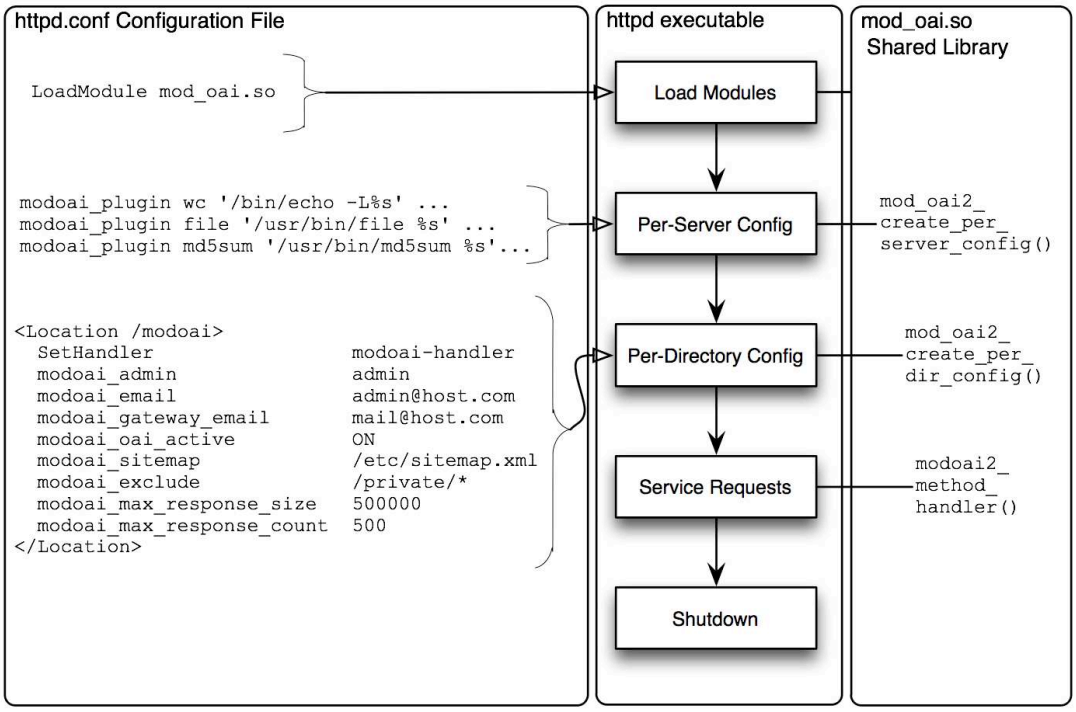


FIG. 67: MODOAI in the Apache web server life cycle.

(e) http://www.foo.edu/**modoai**/?verb=GetRecord&metadataPrefix=crate&identifier=barr.html

Response content generaton for the URL shown in (a) is handled by Apache, but for the URL in (b) it is handled by MODOAI because of the use of "/modoai" in the URL. The addition of "/modoai" to web root as shown in (b) is called the baseURL. Once MODOAI is the handler, the request must follow OAI-PMH syntax rules. Therefore, (d) is an invalid request because it does not have the proper syntax. It will generate a "Bad Request" error from MODOAI, whereas (e) will produce a valid response. Example (c) is a valid Apache request and will receive a "200 OK" response. Further details on MODOAI request processing are provided in Section 4.2 later in this chapter.

2.3 Platform Independence

Apache modules rely extensively on the Apache Portable Runtime to achieve platform independence. APR is an independent Apache project initiated by the Apache development team to isolate the Apache HTTP server from the differences in the underlying operating systems. The MODOAI module depends on APR functions in order to maintain platform independence. That is, by using the native Apache APR functions the module can be compiled and run without modification on a

variety of Unix-based operating system platforms such as Ubuntu Linux and Mac OS X[1]. A few of the key functions MODOAI relies on to maintain this platform independence are described here.

apr_array Arrays are used to hold lists of C style void pointers in a platform independent manner. In MODOAI, apr_arrays hold two important data structures, the list of plugin records (struct oai_plugin_rec) and the list of resources to be processed for a given request. Since apr_arrays do not allow deletion by index number, the pointers are set to NULL to temporarily represent deleted nodes, and then the apr_array is selectively copied to a new, dense array before being returned to the caller.

apr_table These structures contain lists of key-value pairs. The MODOAI code uses apr_tables to store two important sets of data: the list of query parameters from the request URL, and the list of parameters associated with a Resumption Token (spelled "resumptionToken" in the query). For the former, the key fields[2] are *verb, Identifier, metadataPrefix, from, until, set* and *resumptionToken*, as defined in the OAI-PMH specification. For the latter case, the keys are the tokens used to save a client's state: firstUnseenRecord, metadataPrefix, from, until and set. Using apr_tables for these lists provides a straightforward, platform-independent data structure for managing this data.

apr_xml_parser The XML Parser function provides platform-independent DOM-based (Document Object Model) parsing. The MODOAI program uses apr_xml_parser to parse the Sitemap.xml file, and walks the DOM to generate an apr_array of the required resources.

apr_pool Memory management is cleanly handled by the Apache pool mechanism. All dynamically allocated resources are tied to an Apache apr_pool structure. For MODOAI, all resources are tied to the pool with request scope. When servicing of the request is completed, all resources which were allocated in that scope are automatically deallocated by Apache. This frees the MODOAI code from tedious memory management tasks, and ensures leak-free memory management.

One of the other important roles APR use plays in MODOAI relates to the multithreaded environment in which Apache operates. In this case, multithreaded means that Apache typically handles concurrent requests from multiple clients, and thus has more than one possible flow of control. Apache manages the swapping of resources for each of the various threads, ensuring that the correct response is returned to the appropriate request thread.

[1] Even Apache must maintain a separate code base for platforms running on Microsoft's Windows operating system (OS). The core architecture of that OS requires completely different strategies for memory management, resource handling, and other HTTP service provisions. MODOAI has not been tested against the Windows version.

[2] field spelling and capitalization shown as OAI-PMH specification defines them

3 HISTORY OF MODOAI

3.1 Original Scope

The original MODOAI software was developed by several graduate students at Old Dominion University between 2000 and 2003 [108, 136, 138] to address deficiencies in the update semantics of HTTP (cf. Chapter V). MODOAI is an Apache module that implements OAI-PMH functionality and support for complex object metadata formats directly into the Apache web server. The goal of MODOAI is to bring more efficient update semantics to the general web crawling community. Instead of the usual request-response per-resource process that takes place between a crawler and a web server, a single OAI-PMH request can produce one response, listing every resource.

OAI-PMH [99, 182] is the de facto standard for metadata interchange within the digital library community, in part because of its rich semantics. Packages for implementing OAI-PMH repositories for XML files have been described in [82, 176], but they are focused on highly constrained scenarios, not general web content, and they do not integrate directly into the web server. The integration of complex object types into the OAI-PMH model enhances its utility as a website crawling solution [184].

As discussed in Section 4 of Chapter II, the ListRecords and ListIdentifiers verbs are used to obtain a complete list of a repository's items (ListSets enumerates each of the MIME types represented in the repository rather than specific items). ListRecords and ListIdentifiers verbs can also be modified with set (i.e., MIME type) and/or date-range restrictions, making the client update process significantly more efficient. In early MODOAI, the response list for these verbs was built from a file system traversal performed at the time of the request (i.e, it was an extemporaneous response, not cached). In practice, the result of such a traversal may not be a full enumeration of the website's resources, since a file system is not necessarily an accurate picture of the website's contents. Dynamically-generated content and content that integrates multiple resources to produce an über-resource may be omitted from a file-system-based list. Also, insofar as Apache directives affect the response to non-OAI-PMH requests, the MODOAI response could potentially be in disagreement with the equivalent request outside of MODOAI. Despite these shortcomings, there were several advantages with this approach. First, the result set was always synchronized with changes in the underlying file set; any update, deletion or addition to the file system was reflected in the response. Another advantage was that it did not depend on the accuracy of an external file (Sitemap, e.g.) and was therefore not subject to errors introduced by the file creator.

As a prototype, the original MODOAI demonstrated the practicality of using OAI-PMH in a generic (i.e., non-digital-library) web server environment. In particular, update semantics were shown to be significantly more efficient [108, 136] for web crawling. As use of the Sitemaps

Protocol spread, and once it was formally adopted by Google, Yahoo, and MSN, an industry-compatible implementation for MODOAI seemed worthwhile. The additional goal of integrating utility-generated metadata with website resources meant that MODOAI itself had to be overhauled.

3.2 Refactoring and Extending MODOAI Functionality

The MODOAI Codebase

The author took over maintenance of the codebase in 2004, and initially focused on segmenting the code into a more maintainable structure. Formal software engineering processes were introduced, and the project was moved into a version control repository, Subversion, which is also the versioning control system used by the Apache Foundation. The code was refactored from a single file containing all the source code into eighteen source code files organized by structural areas. In addition, functions which could be based on Apache's APR libraries rather than defined as MODOAI-specific functions were converted. This helped resolve some security and memory-management issues and ensured that system resources were properly disposed of at the end of processing.

When taking over existing code, there is a danger of losing existing functionality because of a lack of understanding of the more subtle implementation aspects that previous authors have already solved. To minimize the risk of functional regression, the author implemented a basic function-testing framework written in Perl, which allows rapid execution of many test cases to ensure that new refactoring does not break any existing functionality. This regression testing framework proved to be critically important to the work that followed.

With the regression test framework and basic refactorings in place, the author began extending MODOAI to implement the CRATE functionality. First, the plugin architecture was created, to handle the new metadata extraction utilities. This created an unwanted inconsistency in the code. MODOAI originally provided built-in metadata from HTTP headers ("metadataPrefix=http_header") and Dublin Core ("metadataPrefix=oai_dc"). The new, configurable metadata extraction utilities were handled with the new plugin architecture, while the built-in metadata utilities (HTTP-HEADER, OAI-DC etc) were handled in the old, monolithic fashion. This inconsistency was resolved by refactoring the legacy utilities to be internally defined as plugins. Within the source code, they differ from the new utilities only in that they are not defined in the configuration file, and do not depend on external software. Built-in utilities are completely implemented within the MODOAI program itself. This refactoring greatly simplifies the code, allowing reduction of duplicate functionality. It also simplifies the introduction of new, built-in utilities to the basic MODOAI installation.

Resource Enumeration

Next, the code was extended to specify the list of resources in the site through a Sitemap file, rather than by file system traversal at dissemination time. The adoption of the Sitemaps Protocol by Google, Yahoo and MSN was one factor in the selection of this method for resource enumeration under MODOAI. There were other factors, and these are discussed in Section 7 later in this chapter. The modoai.conf file, which provides module-specific information to Apache, now also includes a parameter specifying the location of the Sitemap (cf. Figure 67).

The next software extension was the *excludes* configuration directive. This directive gives webmasters even more control over resources exposed via MODOAI, allowing specific lists of excluded resources to be enumerated in the Apache configuration file. These excludes take precedence over resources listed in the Sitemap file. The *modoai_exclude* parameter is included in the modoai.conf file. Its syntax follows Unix Regular Expression ("regex") syntax.

Resumption Tokens

An important feature of the OAI-PMH specification is Resumption Token support. Resumption Tokens are issued by the server, and provide a mechanism for clients to retrieve resources in chunks, rather than in one large transaction. This allows a client to not have to reissue the entire request in the case of network errors, and can allow a server to throttle clients based on current load.

The original MODOAI implementation lacked centralized Resumption Token support. Each OAI-PMH verb independently implemented Resumption Token support. Although functional, this made Resumption Token support difficult to test and to maintain.

Resumption Tokens are unique to OAI-PMH, designed to enable flow control on the server side to client requests resulting in large responses. ListSets, ListRecords and ListIdentifiers are examples of verbs where the response size may be larger than the server can normally handle at one time. Throttling of the response is managed at the server by setting Resumption Token values.

The above refactoring allowed the author to rework Resumption Token support, which is now centralized. The original version had used two MODOAI configuration parameters, *modoai_encode_size* and *modoai_resumption_count* which limited a repository's response size by bytes or item count. Because handling of these parameters was completely redefined, they were renamed to *modoai_max_response_count* and *modoai_max_response_size*. Immediately after parsing the Sitemap file, the list of resources is pruned based on the modoai_max_response_count configuration parameter. This happens in a single code location, prior to executing the OAI-PMH verb handling routines.

The other Resumption Token configuration parameter, modoai_max_response_size, specifies the maximum number of characters to issue in the response prior to triggering a Resumption

Token. This cannot be determined until the verb handlers generate the response. To cleanly handle this, each verb handler calls the check_resumption_size() function after each resource is processed, allowing centralized logic for the size parameter. If the size threshold has been exceeded, this function returns a Resumption Token, and no further resources are processed by the verb handler. The resource in question completes processing, however; no partial responses are returned by the server.

4 THE DESIGN & STRUCTURE OF MODOAI

The MODOAI module is written in ANSI C, with external dependencies on the Apache web server and the Apache Portable Runtime. The dependencies on the Apache APR are resolved through the *APache eXtenSion* (apxs) tool. The apxs tool is compiled with, and distributed as a part of, the Apache web server. It exposes information about the compile time settings and file system paths used by the Apache web server. For example, to resolve the location of the installed Apache httpd include files, the MODOAI Makefile queries apxs (indirectly) using the following command line:

```
> apxs -q INCLUDEDIR (command line)
/usr/include/apache2 (response)
```

Structurally, the MODOAI codebase is segregated into 18 ANSI C files, with 18 corresponding header files publishing the public interfaces to MODOAI functions. The main interface files are:

error.c Contains code for generating OAI-PMH error responses. From an Apache perspective (i.e., as far as logging is concerned), all MODOAI responses are HTTP response code 200. This is because the MODOAI module is the handler for the requests, and it returns a response as opposed to failing at the HTTP level. If the request has OAI-PMH errors, the error message is generated by MODOAI returning the error information, not Apache. Thus the response can be both "200 OK" from the Apache perspective and an error from the OAI-PMH perspective.

grammar.c Contains code for checking the grammar of incoming request. The check happens in three parts.

1) *Basic URL validation*: Confirms that the URL contains name/value pairs and that the structure is correct (?verb= not verb=?, e.g.).

2) *OAI-PMH grammar check*: "Grammar" means spelling (ListRecords not listRecords, e.g.), as well as valid elements (for example, the argument is "metadataPrefix", not "metadataFormat" and the verb is "ListMetadataFormats" not "ListMetadataPrefix").

3) *Argument validation*: OAI-PMH verb arguments fall into one of three categories, required, optional, or exclusive. This check confirms that the arguments passed with the verb

are *grammatically* allowed for that particular verb. The Identify verb, for example, cannot have any arguments with it.

This process ensures that MODOAI does not attempt to handle any incoming request which does not conform to valid OAI-PMH grammar. Pruning of resources from the response list due to these arguments occurs later, after the Sitemap file has been read.

mod_oai.c This is the main entry point for all MODOAI interaction with the Apache HTTP server. It contains the callbacks Apache calls when parsing the httpd.conf file, allowing MODOAI to configure itself. It contains the hooks MODOAI uses to register itself with the running Apache HTTP process. It also contains the main callback Apache calls to allow MODOAI to handle the incoming requests during the content generation phase of the request.

response.c Contains helper routines used by the six verb handling routines.

string_util.c Contains helper routines for manipulating strings.

OAI-PMH Verbs code: Contains code for handling each of the six OAI-PMH verbs. This includes both the logic for handling the request and the code for generating the OAI-PMH XML output. Each verb has its own file: (a) vGetRecord.c (b) vListSets.c (c) vListMetadataFormats.c (d) vIdentify.c (e) vListRecords.c (f) vListIdentifiers.c

xml.c Contains code for generating XML snippets which are common to more than one of the verb handlers.

xml_parse.c Contains XML parsing routines for reading the Sitemap.xml file. These routines internally use the apr_xml_parser utilities provided by the APR framework.

execExternal.c Contains code to spawn new processes to run the plugin utilities specified in the httpd.conf file.

handler.c Contains helper code used by the main MODOAI Apache request handling utility.

plugin.c Contains code for configuring and executing both internal (built-in) and external plugin utilities.

resumption.c Contains code for parsing and generating OAI-PMH Resumption Tokens.

timefunc.c Date and time formatting utilities to match the OAI-PMH Guidelines [100].

4.1 Module Configuration & Initialization

The lifetime of an Apache HTTP server, depicted in Figure 66, was discussed briefly in Section 2.1. This section provides a more detailed look at what happens during initialization with

respect to the MODOAI module. At startup, Apache parses the configuration file for all defined servers, which may include one or more virtual servers. For each server, Apache executes all per-server-config callbacks registered by any configured Apache modules. Next, for each directory specified in the httpd.conf file, Apache executes all per-directory-config callbacks registered by any configured Apache modules. At this point, Apache is fully operational and begins accepting requests. For each request having a registered handler module, Apache calls the registered handler callback. For MODOAI, this is `modoai2_method_handler()`. Finally, Apache shuts down and frees all resources.

The MODOAI module has three hooks into the Apache life cycle: (1) at the per-server-config, (2) at the per-directory-config, and (3) a request handler. During the per-server-config processing, MODOAI parses the configuration file for any modoai_plugin lines, and *persistently* configures them into the Apache process space (cf. lines 11–21 of Appendix E–1). At startup, the parser only reads the files for correct syntax, it does not actually verify that the utilities are present or executable. Such checks are more appropriate at request time, as any utilities may change between startup and the time they are called. During the per-directory config processing, MODOAI parses and configures all features not related to the MODOAI plugins. The third callback registered with the Apache HTTP runtime is the content handler, modoai2_method_handler(). This is the function Apache calls when a request arrives which Apache has been configured to pass on to the MODOAI module.

4.2 Request Handling

Apache executes the MODOAI callback modoai2_method_handler() when an OAI-PMH request arrives. Figure 68 depicts the internal processing MODOAI follows to handle the request. The MODOAI module handles the response in seven stages.

Stage 1: First, MODOAI parses the arguments. It ensures the request is a GET request, parses the query string, and does all appropriate "unescaping" of HTTP sensitive characters.

Stage 2: Next, MODOAI performs grammar checks. Grammar checks scan the request arguments to ensure the conform to OAI-PMH syntax. For example, if a verb is provided, it must be one of *Identify, GetRecord, ListRecords, ListIdentifiers, ListSets or ListMetadataFormats*. If any grammatical errors are found, an error page is issued, and no further processing occurs. Note that all parts of an OAI-PMH request are case-sensitive; "getRecord" will generate an error message since the initial "g" was incorrectly lower-case.

Stage 3: The next stage is to check argument validity. This is related to, but distinct from grammar checking. This stage checks the request to ensure it is logically proper. For example, a Resumption Token can only be presented with ListRecords, ListSets and ListIdentifiers verbs. To present a request with a Resumption Token and an Identify verb is not valid. The MODOAI module

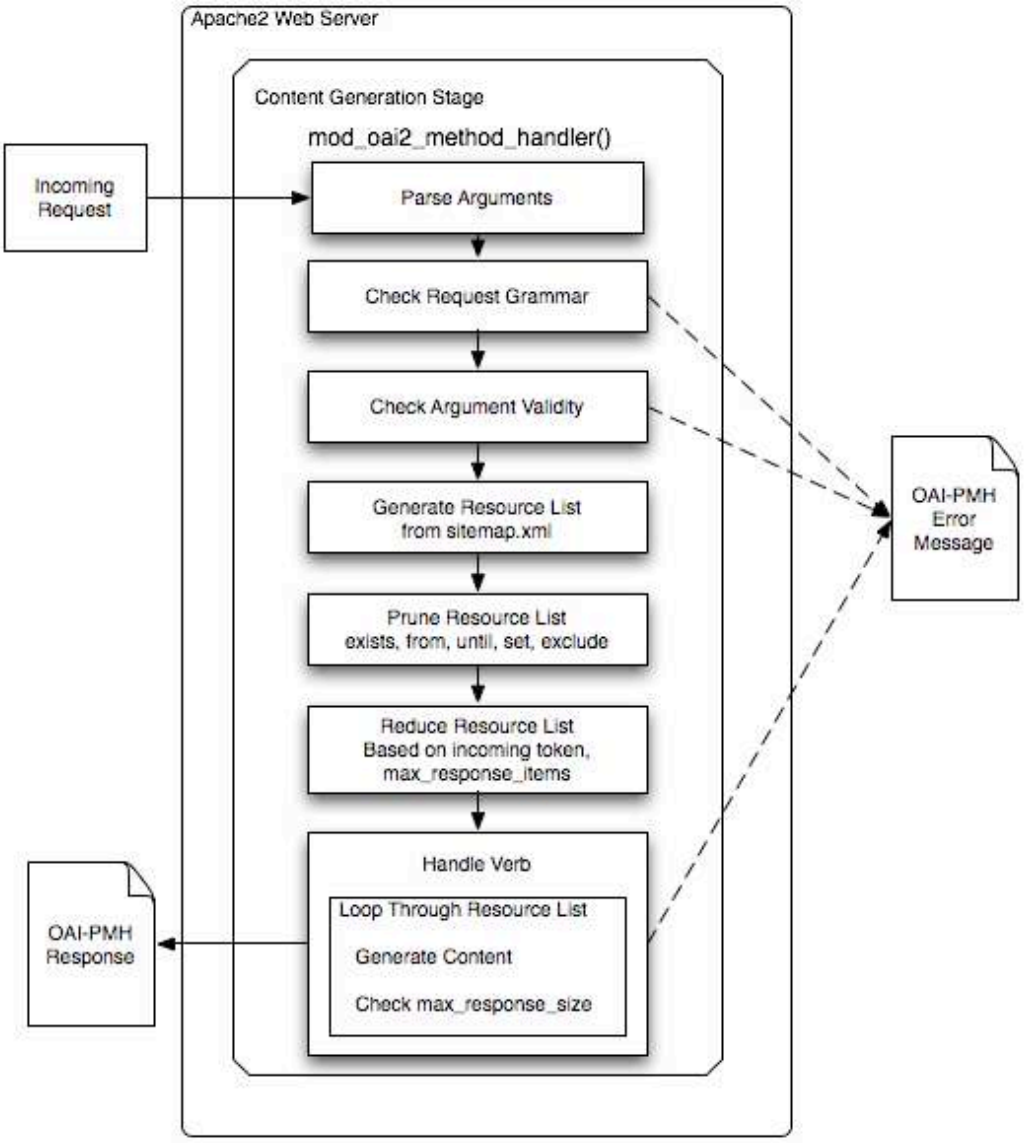


FIG. 68: The MODOAI processing loop.

will issue an error page and stop further processing.

All OAI-PMH requests except Identify and ListSets operate on one or more resources. The fourth, fifth and sixth stage of processing deal exclusively with constructing a list of resources which will be included in the response to the request.

Stage 4: Generate Resource List, reads the Sitemap file and generates a list of all the resources contained in the file.

Stage 5: Prune Resource List, takes the list generated in stage four and reduces it in size based on the following constraints: Any resources which are not on the file system are removed. Any resources which are excluded by a from, until, or set in the query are removed. Any resources covered by a modoai_exclude statement in the httpd.conf file are removed.

Stage 6: Reduce Resource List, is only executed if there is a Resumption Token included in the incoming request. If so, any resources which have previously been sent are excluded. If there are more resources remaining than specified in the modoai_max_request_size parameter, then those are removed. This leaves a resource list consisting of just the resources which should be handled in this request, and possibly an outgoing Resumption Token which should be appended to the end of this request.

Stage 7a - Static Resources: For static resources (i.e., those which reside on the file system), MODOAI is the registered handler. From an execution point of view, this means that the resource passed from the Apache subrequest to MODOAI, which can then direct the content to each of the appropriate metadata utilities if a CRATE metadata format (such as oai_didl or odu_crate) has been specified in the request.

Stage 7b - Dynamic Resources: The Apache API has certain limitations that impact the ability of modules, including MODOAI, to access dynamic content. A module cannot speculatively execute a subrequest for a dynamic resource. Any content generated by that subrequest is immediately passed back to the client and the request terminates without further processing. Under the Apache API, a module is effectively restricted to elements for which it is the registered handler. For dynamic resources, the registered handler may be any of a variety of executable programs. The subrequest in this case is *not* returned to MODOAI but is instead passed directly to the client by the content-handler (i.e., the dynamic module). A work-around for this problem has been introduced which operates as follows:

(1) Generate an external request for the resource

(2) Write the response from this request to a temporary file

(3) Continue processing the response as though it is a static resource

Use of undocumented features can produce problems in future versions of Apache, so MODOAI must handle dynamic resources in a way that does not behave unexpectedly. The temporary file could be streamed directly into the response or into the metadata utilities, but not all utilities will

accept redirection. In these cases, the temporary file method must be used.

Stage 7c - All Requests: The final stage is to actually process the request. For each resource in the final resource list (i.e., after Stage 6), the XML content is generated, and a running total of the number of characters in the response is tallied. After processing a resource, the character total is compared with the configuration parameter modoai_max_response_size. If the parameter has been exceeded, then no further resources are processed. The XML response is completed, and a new Resumption Token is issued. Note that the complete response is sent to the client even if the size limit has been reached. That is, MODOAI will not issue an incomplete response.

5 CUSTOMIZING MODOAI

The MODOAI program can be extended to take advantage of additional metadata extraction utilities through the use of plugins. Plugins are configured in the httpd.conf[3] file, through the modoai_plugin directive. The plugin can be any program which can be executed from the command line. The configuration directive takes four arguments:

```
modoai_plugin zipinfo '/usr/bin/zipinfo %s'
'/usr/bin/zipinfo --version' */zip
```

The first argument of the modoai_plugin line is a label ("zipinfo") which is used to identify the plugin within the output XML file. The second argument is a command-line-executable statement which will generate the plugin output. This command line statement is parsed for the string %s, which is replaced by MODOAI with the name of the file system path to the resource being examined. The third argument is also a command line program to execute, but it is used to provide a version of the utility. For programs which do not have a built in version command, a simple /usr/bin/echo execution statement can be provided to give the appropriate version information. The final argument is a regex pattern which is compared against the MIME-type of the resource to determine if this plugin applies to the resource. For example, the Exif Tool utility examines the metadata of digital photography files. It would not be appropriate to apply this utility to a file with a MIME-type of text/html. For Exif Tool, a configuration could be created which only matches the various JPEG file types, i.e., images where the MIME-type begins with the letters "jp":

```
modoai\_plugin exiftool '/usr/local/bin/exiftool \%s'
'/usr/local/bin/exiftool -v' image/jp*
```

This directive will have Exif Tool operating against both image/jpeg and image/jp2 (the newer, JPEG 2000) MIME types.

[3] Some versions of Apache use a single httpd.conf file. Others use a combination of files, of which modoai.conf would be one.

In some cases, wrapper scripts are helpful in making utilities conform to the above convention. One such utility is Jhove. The Jhove command line requires a specification as to which Jhove module to use for metadata extraction. For example, when processing gif files, the GIF-HUL processor is used, while the PDF-HUL processor is specified on the command line for PDF files. To allow multiple command lines to be called based on the MIME type of the file being passed in, the author wrote a simple perl wrapper which constructs the appropriate Jhove command line based on the MIME type of the resource being examined. For jpeg resources, Jhove is called as:

```
jhove -m jpeg-hul \%s
```

while for GIF files Jhove is invoked as:

```
jhove -m gif-hul \%s
```

(Jhove is not case-sensitive with regard to HUL naming, so these could be written as "GIF-HUL" for example). To use Jhove for only GIF images, the MODOAI configuration line should be written as follows:[4]

```
modoai_plugin jhove "/opt/jhove/jhove -c
/opt/jhove/conf/jhove.conf -m gif-hul -h xml %s"
"/opt/jhove/jhove -c /opt/jhove/conf/jhove.conf
-h xml -v" "image/gif"
```

The MODOAI plugin architecture is designed to make it easy to add more metadata extraction utilities, including other Jhove declarations, so that it is possible to have one for each type of HUL processor desired. Any combination of shell scripts, Perl scripts, etc. can be used to customize MIME or file handling, including handling based on naming conventions or directory location.

6 CRATE DEPLOYED IN MODOAI

One of the goals of the new MODOAI was to enable any type of metadata analysis utility to be incorporated into the software. Since MODOAI is subject to the security rules of the Apache web server, CRATE-type responses can be restricted to authenticated users while still allowing other MODOAI responses to be publicly available. Resumption Tokens allow web servers to further refine the response parameters, enabling them to provide CRATE responses while not impacting overall system performance of standard Apache responses.

Using MODOAI to implement the CRATE model accelerates and simplifies the process of producing preservation-ready web resources. Because it installs and is configured using methods

[4]The example given here is on more than one line for formatting purposes, but it must be on *a single line* in the configuration file.

familiar to most webmasters, it does not impose significant additional workload. An obvious caveat here lies with the metadata utilities themselves. The author found that many were quite simple to install (Jhove, for example, was particularly easy) and a number of others are included as part of most Linux-based OS installations (the Hash functions, Exif Tool and "file" for example). Others required considerable systems expertise (notably Open Text Summarizer and Metadata Extractor). These problems are unlikely to be resolved while such utilities remain in the purview of researchers and archivists rather than in widespread adoption by the average webmaster.

Regardless of the utilities chosen for inclusion, the primary tool which enables CRATE (MODOAI) is just another in a long list of Apache web server modules setup by the webmaster at site initialization. The plugin architecture is simple to customize via the standard configuration files, and a variety of useful utilities (like "file" and Base64 encoding) are likely to have been pre-installed. Many other utilities can be added, and the syntax for inclusion is straightforward even if individual utilities have challenging installation idiosyncracies. One advantage in using MODOAI is that webmasters are familiar with their local resource types and may be better able to select plugin metadata extractors that are appropriate to the site.

The practicality of a CRATE implementation was demonstrated in a series of experiments which were described in Chapter IX. Still, web-server-based preservation is a new concept which may require time before becoming a widely-implemented option at quotidian websites. Open Source projects have had a significant impact on web services (the Apache web server is a good example), and Open Source metadata utilities could eventually play a major role in improving the quality and type of information available for web resource preservation.

7 MODOAI USE OF SITEMAPS

Why Sitemaps

The Counting Problem affects even the best search engines, including Google, Yahoo and MSN. The announcement of the Sitemaps Protocol adoption [146] by the "Big Three" was perhaps inevitable. As discussed in Chapter V, there are limits to blind crawling; enumerating website resources is not trivial. By placing the burden of enumeration on the webmaster, search engines can save bandwidth and processing time. Adopting the Sitemap Protocol also helps MODOAI webmasters.

It has two distinct benefits. First, the webmaster is given very precise control of the resources exposed by MODOAI through the standardized Sitemap format, which can be generated using a wide range of third party tools. Previously, the webmaster depended on the internal MODOAI file system traversal mechanism to select the exposed resources. Short of a manual examination of the MODOAI source code, or a manual sampling of MODOAI query results, webmasters had

no visibility into the exposed resource list. Discrepancies between served resources and listed resources could exist yet the webmaster would be unaware of them.

Second, reading a single Sitemap file has better performance than manually traversing the file system to generate the resource list. File system traversal happens once (if at all), at Sitemap generation time rather than each time a MODOAI request is serviced. If website resources are not stored on the file system, then such a traversal may never occur but instead rely on other methods (e.g., the log analysis method mentioned in Chapter VI). There are tradeoffs, however: Synchronization with the file system's modifications was automatic with the earlier MODOAI prototype whereas a Sitemap must be updated with every change to the underlying resources. Unless the process is fully automated, this can be a time-consuming process at websites that frequently alter their content either by addition or removal. A removed resource will be handled properly by MODOAI, but an additional resource which is not in the Sitemap will be missed. Frequent resource modifications, which impact the result set of the List verbs modified by date range parameters, are correctly handled by MODOAI because it does not depend on Sitemap metadata. Instead, it uses the Sitemap exclusively as a list of URLs and examines each URL as it iterates through the list of resources.

Private VS Public Sitemaps

A new feature of the revised MODOAI is the ability to have a Sitemap which is exposed only to MODOAI clients. This file could be placed in an area not normally accessible by web clients so that it could not be "accidentally" discovered or guessed at by non-MODOAI crawlers. If, for example, it was located in /etc/other/Sitemap.xml, a URL would not usually be able to access it. The MODOAI module has a higher level of system privileges and so can read the file for processing even though its clients cannot. The Sitemap file is not passed to Apache (unlike the Google request for Sitemap.xml, for instance). Rather, it is processed by MODOAI which uses it to build the response list. This is true regardless of whether the Sitemap file is private to MODOAI alone or is the same as the public version available to regular web clients and crawlers.

Sitemaps and Resumption Tokens

Using Sitemaps in MODOAI simplifies Resumption Token processing, making it more efficient. The module walks the Sitemap file until it reaches the Resumption Token point. In early versions of MODOAI, the entire response had to be generated first, and then the Resumption Token point would be calculated. The response would then resume from the calculated point. Subsequent Tokens would invoke the same lengthy creation-calculation sequence. The new method is straightforward and does not involve as much processing.

This approach does introduce a problem, however. If a Sitemap has changed between the prior

Resumption Token and the subsequent request, the response may vary from what was originally anticipated. That is, a response does not necessarily represent the actual Sitemap at any point in time. This problem can be addressed in various ways, but each has its own consequences. First, the Sitemap file could be *versioned* and the Resumption Token could carry the version with it. This approach would still not ensure that a particular response represents a particular Sitemap. A resource "xyz.html" in Sitemap version 1 that was removed in Sitemap version 2 because the resource no longer exists creates a problem. The module prunes resources based on both resource viability (exists, is restricted, etc.) and on OAI-PMH parameters (date range, set membership). Since xyz.html has been removed, MODOAI will skip the item from the version 1 Sitemap. The result is therefore not Sitemap version 1 and may not be representative of Sitemap version 2 either, particularly if new resources are in the Sitemap version 2 file.

Another approach would be to track the Sitemap datestamp and/or version with the Resumption Token and notify the client if the Sitemap has changed since the Token was issued. The server could require the client to regenerate the original request. Depending on the stopping/regeneration point in the Sitemap, this could cause significant extra processing by both client and server. Change is an inherent characteristic of the World Wide Web, its contents are in constant flux. With such an on-going race condition perhaps the best that can be expected is a "good enough" approximation.

Other Considerations

Dynamic and automatic updating of a Sitemap has been a kind of Holy Grail for webmasters. The latest Macintosh operating system, Mac OS X, has the *FSEvents* API [3] that will notify registered applications of any changes to the file system. This could be used to dynamically maintain an always-correct Sitemap.

The MODOAI implementation of Sitemaps has certain advantages over the conventional approach used on the web. First, any errors in the Sitemap file are silently ignored by MODOAI, i.e., no error is sent to the client. If a file is on the list but not found, MODOAI simply skips over it. The Sitemap list is pruned by a process that examines each aspect of a resource before appending it to the response. MODOAI

(1) Reads the entire Sitemap and puts it in a vector
(2) Looks at permissions for each item
(3) Removes the item from the list if it fails the accessibility test
(4) Tests the item for other parameter restrictions like Set membership and datestamp
(5) Removes the item from the list if it fails any of these tests
(6) If the item passes every test then it is appended to the response

The major benefit of this approach is an improved Sitemap delivery to client. Regular Sitemap

access does not do this pruning. A Sitemap may contain restricted, removed, or renamed resources which the search engine crawler will not discover until it has unsuccessfully attempted to request the item. MODOAI webmasters can adjust the logging level of the Apache server to track resources that are pruned due to, for example, unavailability. This could assist the webmaster in maintaining a better, more accurate Sitemap.

8 SUMMARY

The MODOAI software is an Apache web server module written in ANSI C which implements the OAI-PMH protocol. The module is compiled into a shared library, and registers itself with the Apache server as a participant in the content generation stage of Apache request processing. Once registered, all incoming requests to mapped URLs call the MODOAI handler to generate content for the OAI-PMH request. The MODOAI code performs input validation, delegates security checks to the Apache subrequest framework, spawns processes for all configured plugins, and generates the response XML for the client's request.

The reworked MODOAI module now treats built-in metadata extraction utilities in the same manner as webmaster-defined metadata extraction utilities, simplifying code maintenance. Another advantage to this approach is that any utility becomes a candidate for automatic inclusion in the source code. If a metadata utility is accepted as a standard component of an archive-ready resource, that utility can be directly included as part of MODOAI, assuming other elements like licensing are compatible.

The author has refactored the original code, introduced formal software engineering processes, and extended the codebase to implement an extensible plugin architecture. New metadata utilities can be added by webmasters through modifications made to the standard Apache httpd.conf configuration file (cf. Appendix E on page 235). Code regressions are avoided by execution of a new functional test suite prior to all code commits. A private Sitemap file can optionally be implemented for MODOAI clients. Whether the Sitemap is public or private, MODOAI provides improved resource enumeration over the standard Sitemap approach.

CHAPTER XI

FUTURE WORK AND CONCLUSIONS

The outcome of any serious research can only be to make two questions grow where only one grew before. Thorstein Veblen (1857 – 1929)[186]

1 FUTURE WORK

This dissertation presents a number of accomplishments with regard to integrating preservation functionality into the web server but it also suggests other areas that would benefit from further research. These fall into roughly three categories, general investigations, metadata utilities research, and further MODOAI development.

1.1 General Investigations

Shortly after the research for this dissertation was completed, a new OAI project was introduced, the Open Archives Initiative Object Reuse and Exchange (OAI-ORE) [101]. This project "defines standards for the description and exchange of aggregations of Web resources."[1] OAI-ORE addresses the issue of web resources that, while accessed via a single URL, typically include multiple URLs: a web page with many embedded images or a Powerpoint slide show converted to a series of pages with Next/Previous links (and which would otherwise be contained in a single Powerpoint file). In particular, it introduces *Resource Maps* which express the relationship of the components to one another. How should this concept be mapped into CRATE? OAI-ORE resource maps should be investigated for CRATE compatibility, and particularly if additional, preservation-enhancing metadata can be incorporated into the CRATE based on information obtained from the resource map.

There are other non-HTTP formats that could participate in preservation, such as the RSS and ATOM syndication formats. There are subtle differences between the two, and these differences may affect which (if either) of these is compatible with the CRATE model. RSS, for example, does not label the *kind* of content it is sending (plain or escaped XML, e.g.) whereas Atom content is explicitly labelled. [90].

One feature of HTTP is the "Accept-Encoding" field which lets clients specify preferences relating to the format of the response, including language and image type (cf. Table 2 on page 9). In this case, the q-value would indicate to the server that the client is requesting a response where the the resource will be in the CRATE Model format, i.e., an XML document with the metadata

[1] Quotation taken from `http://www.openarchives.org/ore/`

and Base64-encoded resource, as specified in the CRATE XML Schema Document (cf. 2). For example:

```
GET /xyz.html
Accept-Encoding: crate;q=0.9, identity;q=0.8, *;q=0.7
```

By this request, the client indicates that it prefers the CRATE encoding, and if that is not available the default encoding or any others. Which one the client receives will depend upon the configuration of that particular web server.

The author has demonstrated that the web server can produce CRATE objects, but the restoration of such objects to a "future" state where the underlying formats no longer exist has not been explored. A simple restoration from Base64 is trivial (computers routinely encode-decode this format when transporting binary files between systems) and not informative. A scenario where a series of formats is hypothetically "evolved" from current format to future format would provide an interesting test of the CRATE Model and the usefulness of the metadata accompanying the "crated" resources.

1.2 Metadata Utilities Research

Most metadata utilities that were available for use during this research were not developed with this implementation (i.e., MODOAI and CRATE) in mind. Although considerable research into automated metadata generation has been done [70], more work is needed. Further investigations into optimization of metadata utilities would be helpful if web servers are to process resources for preservation metadata. Perhaps more important would be the development of a core set of utilities that would become a default set of metadata elements in any CRATE. With the redesign of MODOAI, "plugging in" additional metadata utilities as *built-in* utilities would greatly simplify the installation and configuration of the module and enhance the quality of preservation metadata in a CRATE.

Another area needing further research which was outside the scope of this dissertation is automated Dublin Core metadata production. The minimal metadata set for OAI-PMH repository interoperability is Dublin Core [98], and a number of *interactive* tools are available online. However, the author was unable to find a working, automated, and command-line-compatible Dublin Core utility that could extract basic information, even from an HTML file with plainly tagged content. The author's home-grown utility relied upon <META> tags in the <HEAD> portion of the HTML document. Google's analysis of such tags [66] has shown that websites rarely implement these tags properly, and perhaps that is the reason that command-line utilities are not in common use at this time. From a web-preservation perspective, this problem is perhaps less urgent than others. From an OAI-PMH interoperability perspective, it is very important because Dublin Core

is the basis for information exchange among repositories [98]. Research and development in automated Dublin Core metadata utilities would benefit CRATE, OAI-PMH, and other preservation and digital library initiatives.

1.3 Further MODOAI Development

Although testing has been performed in a commercial web test environment, further investigations using live websites would be helpful for parameterizing the performance of the CRATE Model and for producing installation guidelines and caveats. In addition to general website testing, investigations into the impact on virtual hosting environments (many websites, 1 machine) may provide further insights into feasibility for such sites and recommendations for improving performance.

The CRATE Model should be implemented in other web server environments (Microsoft's IIS for example) to see if it is practical for these servers. MODOAI produces its XML document output in plain ASCII, which is compliant with the guidelines established for OAI-PMH in the sense that ASCII is a subset of UTF-8, and is backward-compatible. In part this is due to the author conforming to the original version of MODOAI which used plain ASCII to build the response. UTF-8 has become the encoding of choice in the international community. Although MODOAI relies on the UTF-8 compatible Apache APR, it has not been tested against character sets which require UTF-8. The main question involves URL handling, since anything that is actual content on the website is converted to Base64. A series of UTF-8 tests should be created which will point out any deficiencies with regard to handling the expanded character set of such URLs.

As noted in Chapter X, Section 4.2 on page 169, the Apache API passes content directly from the content generator to the client. This operation therefore impacts the ability of modules to access that dynamic content for further processing. Because the dynamic content is not a file system resource but instead is generated by modules other than MODOAI, the other module(s) become the designated handler and the response content bypasses MODOAI, returning directly to the requesting client. On its own, this process would interrupt the OAI-PMH response, closing the connection once the dynamic content was delivered. As a work-around for this issue, as soon as MODOAI identifies a resource as dynamic, it forks a separate request process for the dynamic resource, for which MODOAI then becomes the recipient client. Alternate solutions for this problem should be investigated which would not add the overhead of invoking the additional process.

A basic set of metadata utilities that are useful, practical, and efficient should be collected and provided as an optional part of a MODOAI installation, together with a set of recommendations regarding their applicability by MIME type. This suggestion relates to the recommendations on metadata utility development and inclusion in Section 1.2 on the previous page.

Open Source projects succeed by virtue of the engineering communities that are built around

them. Encouraging development and extension by other programmers will make MODOAI more viable as a software solution for preservation functions in the web server. Proliferation is an important component of success for any Open Source product. The more people that are involved in its development, and the more servers that use it, the more likely it is to succeed. Efforts should be put into assisting adoption of the model at as many sites as possible.

2 CONTRIBUTIONS

This dissertation presents significant theoretical and software contributions to digital preservation, particularly to the preservation of everyday websites. These contributions fall into two main categories, theoretical contributions and software contributions.

2.1 Theoretical Contributions

The CRATE model presented in this dissertation is based on the idea that "pretty good" preservation is a realistic, achievable and worthwhile goal. Just as "Pretty Good Privacy" (PGP) sought to democratize cryptography [203, 202], the CRATE model seeks to democratize digital preservation by placing it in the hands of everyday webmasters, available on web servers anywhere. The theoretical contributions made by this dissertation are:

1) Preservation functions are integrated directly into the web server so that any resource provided by the server can be packaged with preservation metadata.

2) Rather than relying on a formalized relationship between the webmaster and the archivist, this dissertation introduces a model of preservation that is available to everyone. That is, it democratizes preservation tasks by letting any web server fill the role of building the OAIS information package.

3) This dissertation introduces a *data-centric* approach to preservation metadata. It moves away from the traditional model of strict validation at ingestion to *best-effort* metadata at dissemination.

4) Metadata is moved from an ontology-dependent structure to a model where the metadata is *undifferentiated*.

5) This dissertation defines and describes the CRATE Model which packages a resource and its preservation information together as a *Complex Object*. A CRATE consists of 3 elements: (i) A UID, (ii) Metadata, and (iii) Resource encoded in Base64. The Model specifies that all CRATE content must be in plain text, considered the lowest-common-denominator encoding which is likely to survive over the long-term. The CRATE Schema is also presented, and examples of a CRATE implementation are provided.

6) As part of the CRATE investigations, this dissertation also documents the behavior of web crawlers. It details patterns that appear to be dependent on website characteristics, and documents the impact of website structure on web crawler behavior.

2.2 Software Contributions

The theory proposed by this dissertation, i.e., that preservation functions can be integrated into the web server, has an applied component. This dissertation presents the first known implementation of preservation functionality integrated into the web server. A number of software contributions are made in this dissertation:

1. A completely revised implementation of OAI-PMH was developed to implement the CRATE Model. Based on an earlier prototype, the version of MODOAI presented in this dissertation has been redesigned to conform with the Apache API.

2. The dissertation introduces an *extensible* architecture to the MODOAI module which enables a variety of metadata utilities to serve as "plugins". Each plugin can be applied to resources individually utilizing a well-understood, Regular-Expressions oriented convention in the web server configuration file.

3. MODOAI provides a foundation for additional preservation functionality to be integrated into the web server.

4. A suite of functional-unit tests were developed together with a test website which can be used to confirm proper installation and operation of the software without affecting other sites hosted by the server.

5. Formal software engineering processes were introduced, and the project was moved into a version control repository, Subversion, which is the versioning control system used by the Apache Software Foundation.

6. The software is multi-platform compatible. As an Apache 2.x built-in module, MODOAI inherits the advantages of the Apache web server. Native Apache calls make the software portable to more than one platform, eliminating the need to maintain multiple development branches. Regardless of the target installation's operating system, modules are handled in a consistent, well-understood manner; the source code does not need insight into the specifics of the target platform. In addition, issues like memory management (important for threaded applications like Apache) are handled by the web server rather than the module.

7. Version 1.0 of MODOAI has been released to the public under a GPL-2 license and is available through Google Code (`http://code.google.com/modoai`) and at the MODOAI

website (`http://www.modoai.org/`).

3 CONCLUSION: INTEGRATING PRESERVATION FUNCTIONS INTO THE WEB SERVER

If not for the Internet Archive's efforts to store periodic snapshots of the web, many websites would not have any preservation prospects at all. The barrier to entry is too high for everyday websites, which may have skilled webmasters managing them, but which lack skilled archivists to preserve them. Digital preservation is not easy. One problem is the complexity of preservation models, which have specific metadata and structural requirements. Another problem is the time and effort it takes to properly prepare digital resources for preservation in the chosen model.

This dissertation presents a novel idea: that the web server which produces and delivers resources can also provide preservation metadata for them. Implicit in this idea is that it is easier to restore a web resource which has preservation metadata packaged with it than one which has none. Experiments implemented as part of this research show that such functionality *can* be incorporated at the web server, and that third-party metadata utilities *can* function within that environment to produce a highly-descriptive complex object consisting of the resource packaged *with* its metadata: an object called a CRATE. The CRATE Model does not require insight into preservation technologies by the webmaster nor the client, and as such it moves preservation from the hands of specialists into the realm of the casual user.

The target for this model is the everyday, personal or community website where a long-term preservation strategy does not yet exist, but the approach could also be effectively applied to semi-formal collections like college course websites or departmental file systems. Even though the utilities to add such preservation metadata are installed on the web server, the preservation point of view is from the *client* side of the website, because that is what website users experience. A crawlable web resource is preservable, but the web server components which created it may not be. Websites could benefit from the democratization of preservation functionality in this nearly-transparent, server-implemented solution. The CRATE Model is presented as an enabling technology in making this transformation possible.

REFERENCES

[1] Amazon, Inc.: Kindle official site. http://kindle.amazon.com/

[2] Apache HTTP server version 2.0 documentation. The Apache Software Foundation (2008). http://httpd.apache.org/docs/2.0/

[3] Apple, Inc.: IPC and Notification Mechanisms, FSEvents API, chap. 2, pp. 36–37. Apple, Inc. (2007). http://developer.apple.com/documentation/MacOSX/Conceptual/OSX_Technology_Overview/OSX_Technology_Overview.pdf

[4] Archive it. http://www.archive-it.org/

[5] Arms, W.Y.: Key concepts in the architecture of the digital library. D-Lib Magazine **1** (1995). http://www.dlib.org/dlib/July95/07arms.html

[6] Arms, W.Y.: Preservation of scientific serials: Three current examples. Journal of Electronic Publishing **5**(2) (1999). http://hdl.handle.net/2027/spo.3336451.0005.202

[7] Arms, W.Y.: Digital Libraries. MIT Press (2000)

[8] Auditmypc.com: Webmaster tool. Java servlet for Sitemap generation (2008). http://www.auditmypc.com/xml-sitemap.asp

[9] Autositemap.com: Online sitemap tool (2008). http://www.autositemap.com/

[10] Baeza-Yates, R., Castillo, C., Efthimiadis, E.N.: Characterization of national web domains. ACM Transactions on Internet Technology TOIT **7**(2) (2007). doi: http://doi.acm.org/10.1145/1239971.1239973

[11] Bar-Yossef, Z., Keidar, I., Schonfeld, U.: Do not crawl in the DUST: Different URLs with Similar Text. In: Proceedings of the 16th International World Wide Web Conference WWW'07, pp. 111–120 (2007)

[12] Bearman, D.: Reality and chimeras in the preservation of electronic records. D-Lib Magazine **5**(4) (1999). http://www.dlib.org/dlib/april99/bearman/04bearman.html

[13] Bekaert, J., De Kooning, E., Van de Sompel, H.: Representing digital assets using MPEG-21 Digital Item Declaration. International Journal on Digital Libraries **6**(2), 159–173 (2006). doi:10.1007/s00799-005-0133-0

[14] Bekaert, J., Van de Sompel, H.: A standards-based solution for the accurate transfer of digital assets. D-Lib Magazine **11**(6) (2005). doi:10.1045/june2005-bekaert

[15] Bent, L., Rabinovich, M., Voelker, G.M., Xiao, Z.: Characterization of a large web site population with implications for content delivery. In: WWW'04: Proceedings of the 13th International Conference on World Wide Web, vol. 9, pp. 522–533. ACM (2004). http://doi.acm.org/10.1145/988672.988743

[16] Bergman, M.K.: The deep web: Surfacing hidden value. Journal of Electronic Publishing **7**(1) (2001). http://www.press.umich.edu/jep/07-01/bergman.html

[17] Berners-Lee, T.: Uniform resource identifiers in WWW: A unifying syntax for the expression of names and addresses of objects on the network as used in the World Wide Web. IETF RFC 1630 (1994). http://www.ieft.org/rfc/rfc1630.txt

[18] Berners-Lee, T., Fielding, R., Frystyk, H.: Hypertext Transfer Protocol – HTTP/1.0. http://www.ietf.org/rfc/rfc1945.txt (1996)

[19] Berners-Lee, T., Fielding, R., Masinter, L.: Uniform resource identifier (URI): Generic syntax. The Internet Engineering Task Force RFC 3986 (2005). http://www.ietf.org/rfc/rfc3986.txt

[20] Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5), 34–43 (2001)

[21] Berners-Lee, T., Masinter, L., McCahill, M.: Uniform Resource Locators (URL). IETF RFC 1738 (1994). http://www.ietf.org/rfc/rfc1738.txt

[22] Bharat, K., Broder, A.: Mirror, mirror on the Web: A study of host pairs with replicated content. WWW8 Computer Networks **31**(11-16), 1579–1590 (1999)

[23] Black, P.E.: multigraph. In: P.E. Black (ed.) Dictionary of Algorithms and Data Structures [online]. U.S. National Institute of Standards and Technology (2006). http://www.nist.gov/dads/HTML/multigraph.html

[24] Borenstein, N., Freed, N.: MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. http://www.ietf.org/rfc/rfc1521.txt (1993)

[25] Bowman, C., Danzig, P., Hardy, D., Manber, U., Schwartz, M.: The harvest information discovery and access system. Computer Networks and ISDN Systems **28**(1-2), 119–125 (1995)

[26] Bowman, C.M., Danzig, P.B., Hardy, D.R., Manber, U., Schwartz, M.F.: The harvest information discovery and access system. In: Proceedings of the Second International WWW Conference '94: Mosaic and the Web (1994). http://portal.acm.org/citation.cfm?coll=GUIDE\&dl=GUIDE\&id=217463

[27] Braden, R.: Requirements for internet hosts – communication layers. IETF RFC 1122 (1989). http://www.ietf.org/rfc/rfc1122.txt

[28] Brand, S.: Far forward. Report from the RLG OCLC Annual Meeting (2000). http://digitalarchive.oclc.org/da/ViewObjectMain.jsp;jsessionid=84ae0c5f82405355a4cd14ef4b89b948a6af86f3bee0?fileid=0000070504:000006276583&reqid=7

[29] Brandman, O., Cho, J., Garcia-Molina, H., Shivakumar, N.: Crawler-friendly web servers. SIGMETRICS Perform. Eval. Rev. **28**(2), 9–14 (2000). http://doi.acm.org/10.1145/362883.362894

[30] Broad, W.J.: Web archive is said to reveal a nuclear primer. The New York Times Online (2006). http://www.nytimes.com/2006/11/03/world/middleeast/03cnd-documents.html?_r=2&bl=&ei=5087%0A&en=e9c8b8a6f22cf4e9&ex=1163134800&adxnnl=1&adxnnlx=1163127354-bgbDa8lKoy7L/9p1OjiGDw&oref=slogin&oref=slogin

[31] Castillo, C.: Cooperation schemes between a web server and a web search engine. In: LA-WEB, pp. 212–213. IEEE CS Press (2003). citeseer.ist.psu.edu/castillo03cooperation.html

[32] Caverlee, J., Liu, L., Buttler, D.: Probe, cluster, and discover: Focused extraction of qa-pagelets from the deep web. In: ICDE, pp. 103–115 (2004). http://csdl.computer.org/comp/proceedings/icde/2004/2065/00/20650103abs.htm

[33] CCSDS: Reference model for an open archival information system ISO 14721:2002. Tech. Rep. CCSDS 650.0-B-1, Consultative Committee for Space Data Systems (2002)

[34] Chan, L.M.: Inter-indexer consistency in subject cataloging. Information Technology and Libraries **8**(4), 349–358 (1989)

[35] Chen, M., Sun, J.T., Zeng, H.J., Lam, K.Y.: A practical system of keyphrase extraction for web pages. In: CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management, pp. 277–278. ACM (2005). http://doi.acm.org/10.1145/1099554.1099625

[36] Cherkasova, L., Karlsson, M.: Dynamics and evolution of web sites: Analysis, metrics, and design issues. In: ISCC, pp. 64–71. IEEE Computer Society (2001)

[37] Cho, J., Garcia-Molina, H.: Effective page refresh policies for web crawlers. ACM Trans. Database Syst. **28**(4), 390–426 (2003). http://doi.acm.org/10.1145/958942.958945

[38] Cho, J., Garcia-Molina, H.: Estimating frequency of change. ACM Transactions on Internet Technology **3**(3), 256–290 (2003). http://doi.acm.org/10.1145/857166.857170

[39] Cho, J., Garcia-Molina, H., Page, L.: Efficient crawling through url ordering. Computer Networks and ISDN Systems **30**(1-7), 161–172 (1998). http://dx.doi.org/10.1016/S0169-7552(98)00108-1

[40] Cho, J., Shivakumar, N., Garcia-Molina, H.: Finding replicated web collections. In: SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 355–366. ACM Press (2000). http://doi.acm.org/10.1145/342009.335429

[41] Cohen, D.: Rough start for digital preservation. Dan Cohen's Digital Humanities Blog (2006). http://www.dancohen.org/2006/01/02/

[42] Commission on Physical Sciences, Mathematics, and Applications: Study on the Long-term Retention of Selected Scientific and Technical Records of the Federal Government: Working Papers (1995), p. 26. National Academies Press (1995). http://darwin.nap.edu/books/NI000157/html/26.html

[43] Connexion Integrated Cataloging Service. OCLC, Online Computer Library Center. http://www.oclc.org/connexion/

[44] Cothey, V.: Web-crawling reliability. Journal of the American Society for Information Scienceand Technology **55**(14), 1228–1238 (2004). doi:10.1002/asi.20078

[45] Cutts, M.: Dynamic URLs and crawlers (2007). http://blog.searchenginewatch.com/blog/070815-092740

[46] DASL: DAV Searching and Locating. http://www.webdav.org/dasl/ (accessed on 3/17/2006)

[47] The Dublin Core Metadata Initiative. History of the Dublin Core Metadata Initiative. http://dublincore.org/about/history/

[48] Dvdxcopy.com: DVD Copy software reviews. http://www.dvdxcopy.com/default.asp

[49] European Archive: Living web archives project. http://www.europarchive.org/liwa.php

[50] Everard, J.: Meta tags – so 200BCE (2007). http://lostbiro.com/blog/wp-content/uploads/2007/03/RosettaStone.jpg

[51] Fetterly, D., Manasse, M., Najork, M., Wiener, J.L.: A large-scale study of the evolution of web pages. Software: Practice & Experience **34**(2), 213–237 (2004)

[52] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. IETF RFC 2616 (1999). http://www.ietf.org/rfc/rfc2616.txt

[53] Fielding, R.T.: Architectural styles and the design of network-basedsoftware architectures. Ph.D. thesis, University of California at Irvine (2000)

[54] Freed, N., Borenstein, N.: Multipurpose internet mail extensions (MIME) Part One: Format of Internet Message Bodies. IETF RFC 2045 (1996). http://tools.ietf.org/html/rfc2045

[55] Freed, N., Borenstein, N.: Multipurpose internet mail extensions (MIME) Part Two: Media Types. IETF RFC 2046 (1996). http://tools.ietf.org/html/rfc2046

[56] Friedlander, A.: Summary of findings. In: Building a National Strategy for Preservation: Issues in Digital Media Archiving. Council on Library and Information Resources and Library of Congress (Co-Publishers) (2002). http://www.clir.org/pubs/reports/pub106/summary.html

[57] Frontiernet: NIST Logo conversion (2008). http://www.frontiernet.net/~imaging/sc_nist_logo_java3d.jpg

[58] Furrie, B.: Understanding MARC Bibliographic: Machine-Readable Cataloging. Library of Congress (2003)

[59] Gibson, D., Punera, K., Tomkins, A.: Volume and evolution of web page templates. In: 14th International World Wide Web Conference, pp. 830–839. ACM, New York, NY, USA (2005). http://www2005.org/cdrom/docs/p830.pdf

[60] Gladney, H.M.: Trustworthy 100-year digital objects: Evidence after every witness is dead. ACMTransactions On Information Systems **22**(3), 406–436 (2004)

[61] Goland, Y., Whitehead, E., Faizi, A., Carter, S., Jensen, D.: HTTP extensions for distributed authoring – WEBDAV. Tech. Rep. Internet RFC-2518, IETF (1999)

[62] Google, Inc: 20 year archive on google groups. http://www.google.com/googlegroups/archive_announce_20.html

[63] Google, Inc: First mention of MS-DOS. Posting by "Frank" (1981). http://groups.google.com/group/fa.info-cpm/msg/c86239309bfbdc31

[64] Google, Inc: World Wide Web - Executive Summary. Posting by Tim Berners-Lee (1991). http://groups.google.com/group/alt.hypertext/msg/395f282a67a1916c

[65] Google, Inc: Creating google sitemaps files. http://www.google.com/support/webmasters/bin/topic.py?topic=8467 (2006)

[66] Google, Inc: Google code: Web authoring statistics. http://code.google.com/webstats/ (2006). Accessed on 10 Feb 2007

[67] Google, Inc: Sitemap tool. Python tool for sitemap generation (2008). https://www.google.com/webmasters/tools/docs/en/sitemap-generator.html

[68] Google, Inc: Webmaster guidelines (2008). http://www.google.com/support/webmasters/bin/answer.py?answer=35769

[69] Granneman, S.: The perils of googling (2004). http://www.theregister.co.uk/2004/03/10/the_perils_of_googling/

[70] Greenberg, J., Spurgin, K., Crystal, A.: Final report for the AMeGA (automatic metadata generation applications) project. Tech. rep., University of North Carolina at Chapel Hill School of Information and Library Science (2005)

[71] Grimaldi, J.V., Eilperin, J., Weisman, J.: Some say they felt uneasy about representative's attention. The Washington Post p. Page A01 (2006)

[72] Guenther, R.S.: MODS: The Metadata Object Description Schema. portal: Libraries and the Academy **3**(1), 137–150 (2003)

[73] Gulli, A., Signorini, A.: The indexable web is more than 11.5 billion pages. WWW 2005 **5** (2005)

[74] Hardy, D.R., Schwartz, M.F.: Customized information extraction as a basis for resource discovery. ACM Trans. Comput. Syst. **14**(2), 171–199 (1996). http://doi.acm.org/10.1145/227695.227697

[75] Henry A. Murray Research Archive. The Institute for Quantitative Social Science at Harvard University. http://www.murray.harvard.edu/frontpage

[76] Harvest: A distributed search system. Sourceforge Project. http://harvest.sourceforge.net/

[77] Hauben, M., Hauben, R.: Netizens: On the History and Impact of Usenet and the Internet. Wiley IEEE Computer Society Press (1997). ISBN 0-8186-7706-6

[78] Hemenway, K., Calishain, T.: Spidering Hacks, first edn. O'Reilly Media, Inc. (2003)

[79] Henzinger, M.: Hyperlink analysis for the Web. IEEE Internet Computing **5**(1), 45–50 (2001)

[80] Heydon, A., Najork, M.: Mercator: A scalable, extensible web crawler. In: WWW '99: Proceedings of the 8th International Conference on World Wide Web, pp. 219–229. Toronto, Canada (1999). http://dx.doi.org/10/1023/A:1019213109274

[81] Hirai, J., Raghavan, S., Garcia-Molina, H., Paepcke, A.: WebBase: a repository of web pages. Computer Networks (Amsterdam, Netherlands: 1999) **33**(1–6), 277–293 (2000). citeseer.ist.psu.edu/article/hirai99webbase.html

[82] Hochstenbach, P., Jerez, H., Van de Sompel, H.: The OAI-PMH static repository and static repository gateway. In: JCDL '03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 210–217. IEEE Computer Society, Washington, DC, USA (2003)

[83] Holtman, K., Mutz, A.: Transparent content negotiation in HTTP. The Internet Engineering Task Force RFC 2295 (1998). http://www.ietf.org/rfc/rfc2295.txt

[84] Hubbard, D.W.: How To Measure Anything. John Wiley & Sons (2007)

[85] (IANA), I.A.N.A.: MIME Media Types. http://www.iana.org/assignments/media-types/ (2007)

[86] International Conference on Preservation of Digital Objects. http://ipres.library.cornell.edu/

[87] Ipeirotis, P.G., Gravano, L., Sahami, M.: Probe, count, and classify: Categorizing hidden web databases. In: SIGMOD Conference, pp. 100–109 (2001). http://doi.acm.org/10.1145/375663/375671

[88] Jamin, S., Jin, C., Kurc, A., Raz, D., Shavitt, Y.: Constrained mirror placement on the Internet. INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings **1** (2001)

[89] Jaschik, S.: University presses start to sell via Kindle. Inside Higher Ed (2008). http://insidehighered.com/news/2008/06/24/kindle

[90] Johnson, D.: RSS and Atom in Action: Web 2.0 Building Blocks. Manning Publications (2006)

[91] Josefsson, S.: The base16, base32, and base64 data encodings. IETF RFC 4648 (2006). http://www.ietf.org/rfc/rfc4648.txt

[92] Kahn, R., Wilensky, R.: A framework for distributed digital object services. Corporation for National Research Initiatives hdl:cnri.dlib/tn95-01 (1995). http://www.cnri.reston.va.us/home/cstr/arch/k-w.html

[93] Kahn, R., Wilensky, R.: A framework for distributed digital object services. International Journal On Digital Libraries **6**(2), 115–123 (2006)

[94] KEA automatic keyphrase extraction tool. New Zealand Digital Library, University of Waikato (2006). http://www.nzdl.org/Kea/

[95] Klein, A.: The insecure indexing vulnerability. Web Application Security Consortium (2005). http://www.webappsec.org/projects/articles/022805.shtml

[96] Klein, M., McCown, F., Smith, J.A., Nelson, M.L.: How Much Preservation Do I Get If I Do Absolutely Nothing?, pp. 71 – 87. GITO-Verlag, Berlin, Germany (2007)

[97] Koninklijke bibliotheek. (National Library of the Netherlands). http://www.kb.nl/

[98] Lagoze, C., de Sompel, H.V., Nelson, M.L., Warner, S.: The Open Archives Initiative Protocol for Metadata Harvesting. http://www.openarchives.org/OAI/openarchivesprotocol.html

[99] Lagoze, C., Van de Sompel, H.: The Open Archives Initiative: building a low-barrier interoperability framework. In: JCDL '01: Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 54–62. ACM Press, New York, NY, USA (2001). http://doi.acm.org/10.1145/379437.379449

[100] Lagoze, C., Van de Sompel, H., Nelson, M.L., Warner, S.: Implementation guidelines for the Open Archives Initiative Protocol for Metadata Harvesting. http://www.openarchives.org/OAI/2.0/guidelines.htm (2005)

[101] Lagoze, C., Van de Sompel, H., Nelson, M.L., Warner, S., Sanderson, R., Johnston, P.: Object re-use & exchange: A resource-centric approach. Tech. Rep. arXiv:0804.2273v1 [cs.DL], Cornell University (2008)

[102] Langer, A.: The post man always saves twice. The Austin Chronicle (1997). http://weeklywire.com/ww/07-14-97/austin_screens_feature1.html

[103] Leonard, L.E.: Inter-indexer consistency studies, 1954-1975:a review of the literature and summary of study results. Graduate School of Library Science, University of Illinois, Urbana-Champaign, IL (1977)

[104] Levering, R., Cutler, M.: The portrait of a common html web page. In: DocEng '06: Proceedings of the 2006 ACM symposium on Document engineering, pp. 198–204. ACM, New York, NY, USA (2006). http://doi.acm.org/10.1145/1166160.1166213

[105] Levy, D.M.: Heroic measures: reflections on the possibility and purpose of digital preservation. In: DL '98: Proceedings of the Third ACM Conference on Digital Libraries, pp. 152–161. ACM Press, New York, NY, USA (1998). http://doi.acm.org/10.1145/276675.276692

[106] Library of Congress: The State of Digital Preservation: An International Perspective. Council on Library Information Resources (CLIR), Washington, DC (2002)

[107] Liu, X., Balakireva, L., Van de Sompel, P.H.H.: File-based storage of digital objects and constituent datastreams: XMLtapes and internet archive ARC files. In: 9th European Conference on Research and Advanced Technologyfor Digital Libraries (ECDL 2005), pp. 254–265 (2005)

[108] Liu, X., Maly, K., Zubair, M., Nelson, M.L.: Repository synchronization in the OAI framework. In: JCDL '03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 191–198. IEEE Computer Society, Washington, DC, USA (2003)

[109] Liu, Z., Luo, C., Cho, J., Chu, W.W.: A probabilistic approach to metasearching with adaptive probing. In: ICDE, pp. 547–559 (2004). http://doi.ieeecomputersociety.org/10.1109/ICDE.2004.1320026

[110] Lyman, P.: Appendix 2: Archiving the World Wide Web, pp. 53–66. Library of Congress and the Council on Library and Information Resources (CLIR) (2002). http://www.digitalpreservation.gov/library/pdf/ndiipp_appendix.pdf

[111] Lyman, P., Varian, H.R., Charles, P., Good, N., Jordan, L.L., Pal, J.: How much information? 2003. Research Project Report, U.C. Berkeley School of Information Management and Systems (2003). http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/

[112] Lynch, C.: Canonicalization: A fundamental tool to facilitate preservation and management of digital information. D-Lib Magazine **5**(9), 1082–9873 (1999)

[113] Lynch, C.: Metadata harvesting and the Open Archives Initiative. ARL Bi-Monthly Report (2001). http://www.arl.org/newsltr/217/mhp.html

[114] Lynch, C.: When documents deceive: Trust and provenance as new factors for information retrieval in a tangled web. Journal of the American Society for Information Science and Technology **52**(1), 12–17 (2001). http://dx.doi.org/10.1002/1532-2890(2000)52:1<12::AID-ASI1062>3.3.CO;2-M

[115] Mackay, A., (Editors), M.E.: The Harvest of a Quiet Eye. Institute of Physics Publishing (1976). ISBN: 0854980318

[116] Maly, K., Nelson, M.L., Zubair, M.: Smart objects, dumb archives. D-Lib Magazine (1999). http://www.dlib.org/dlib/march99/maly/03maly.html

[117] Maniatis, P., Roussopoulos, M., T.J.Giuli, Rosenthal, D.S.H., Baker, M.: The LOCKSS peer-to-peer digital preservation system. ACM Transactions on computer systems **23**(1), 2 – 50 (2005)

[118] March, S.T., Scudder, G.D.: On the selection of efficient record segmentations and backup strategies for large shared databases. ACM Transactions on Database Systems **9**(3), 409–438 (1984). http://doi.acm.org/10.1145/1270.1481

[119] McCown, F., Benjelloun, A., Nelson, M.L.: Brass: A queueing manager for warrick. In: 7th International Web Archiving Workshop (IWAW 2007) (2007)

[120] McCown, F., Diawara, N., Nelson, M.L.: Factors affecting website reconstruction from the web infrastructure. In: JCDL'07: Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries, pp. 39 – 48 (2007). doi:10.1145/1255175.1255182

[121] McCown, F., Liu, X., Nelson, M.L., Zubair, M.: Search engine coverage of the OAI-PMH corpus. IEEE Internet Computing **10**(2), 66–73 (2006)

[122] McCown, F., Smith, J.A., Nelson, M.L., Bollen, J.: Reconstructing websites for the lazy webmaster. Tech. Rep. arXiv:cs.IR/0512069, Old Dominion University (2005). http://arxiv.org/abs/cs.IR/0512069

[123] McCown, F., Smith, J.A., Nelson, M.L., Bollen, J.: Reconstructing websites for the lazy webmaster. Proceedings of the eighth ACM international workshop on web information and data management (WIDM) pp. 67–74 (2006). http://doi.acm.org/10.1145/1183550.1183564

[124] McDonough, J.P.: METS: Standardized encoding for digital library objects. International Journal on Digital Libraries **6**(2), 148–158 (2006). 10.1007/s00799-005-0132-1

[125] Miller, E.: An introduction to the resource description framework. D-Lib Magazine **4**(5) (1998). http://dlib.org/dlib/may98/miller/05miller.html

[126] Mishne, G., Carmel, D., Lempel, R.: Blocking blog spam with language model disagreement. In: Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web AIR Web '05 (2005)

[127] Mohr, G., Kimpton, M., Stack, M., Ranitovic, I.: Introduction to heritrix, an archival quality web crawler. In: Proceedings of the 4th International Web Archiving Workshop (IWAW '04) (2004)

[128] Mueller, J.: Retiring support for OAI-PMH in sitemaps. Google Webmaster Central Blog (2008). http://googlewebmastercentral.blogspot.com/2008/04/retiring-support-for-oai-pmh-in.html

[129] National Archives Digital Preservation Department (UK). http://www.nationalarchives.gov.uk/preservation/digital.htm

[130] National Library of Australia Digital Services Project. http://www.nla.gov.au/nla/staffpaper/dw001004.html

[131] NDIIPP, the National Digital Information Infrastructure and Preservation Program. http://www.digitalpreservation.gov/

[132] Nelson, M.L., Argue, B., Efron, M., Denn, S., Pattuelli, M.C.: A survey of complex object technologies for digital libraries. Tech. Rep. NASA/TM-2001-211426, NASA Langley Research Center (2001). http://techreports.larc.nasa.gov/ltrs/PDF/2001/tm/NASA-2001-tm211426.pdf

[133] Nelson, M.L., Bollen, J., Manepalli, G., Haq, R.: Archive ingest and handling test, the Old Dominion University approach. D-Lib Magazine **11**(12) (2005). doi:10.1045/december2005-nelson

[134] Nelson, M.L., McCown, F., Smith, J.A., Klein, M.: Using the web infrastructure to preserve web pages. International Journal on Digital Libraries **6**(4), 327–349 (2007). doi:10.1007/s00799-007-0012-y

[135] Nelson, M.L., Smith, J.A., Klein, M.: Repository replication using smtp and nntp. In: dg.o '06: Proceedings of the 2006 International Conference on Digital Government Research, pp. 436–437. New York, NY, USA (2006). http://doi.acm.org/10.1145/1146598.1146737

[136] Nelson, M.L., Smith, J.A., Van de Sompel, H., Liu, X., Garcia del Campo, I.: Efficient, automatic web resource harvesting. In: Proceedings of the seventh ACM international workshop on web information and data management, pp. 43–50 (2006). http://doi.acm.org/10.1145/1183550.1183560

[137] Nelson, M.L., Van de Sompel, H.: IJDL special issue on complex digital objects: Guest editors' introduction. International Journal on Digital Libraries **6**(2), 113–114 (2006). Doi:10.1007/s00799-005-0127-y

[138] Nelson, M.L., Van de Sompel, H., Liu, X., Harrison, T.L.: mod_oai: An apache module for metadata harvesting. Tech. Rep. arXiv:cs/0503069v1, Old Dominion University (2005). http://arxiv.org/abs/cs.DL/0503069

[139] Netcraft server survey. http://news.netcraft.com/archives/web_server_survey.html

[140] NISO (National Information Standards Organization): Understanding Metadata. NISO Press (2004). http://www.niso.org/publications/press/UnderstandingMetadata.pdf

[141] Ntoulas, A., Cho, J., Olston, C.: What's new on the web?: the evolution of the web from a search engine perspective. In: WWW '04: Proceedings of the 13th International Conference on the World Wide Web, pp. 1–12 (2004). http://doi.acm.org/10.1145/988672.988674

[142] Ntoulas, A., Najork, M., Manasse, M., Fetterly, D.: Detecting spam web pages through content analysis. In: WWW'06: Proceedings of the 15th International Conference onf World Wide Web (2006)

[143] Ntoulas, A., Zerfos, P., Cho, J.: Downloading textual hidden web content through keyword queries. In: JCDL '05: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 100–109 (2005). http://doi.acm.org/10.1145/1065385.1065407

[144] O'Neill, E.T., Lavoie, B.F., Bennett, R.: Trends in the evolution of the public web. D-Lib Magazine **3**(4) (2003). http://dx.doi.org/10.1045/april2003-lavoie

[145] Pant, G., Srinivasan, P., Menczer, F.: Crawling the Web, chap. II, pp. 153 – 178. Springer-Verlag (2004)

[146] Perez, J.C.: Google, Yahoo and Microsoft partner to help webmasters. Computerworld (2006). http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9005148

[147] Pilgrim, M.: What is rss. O'Reilly XML.com (2002). http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html

[148] Postel, J.: DOD Standard internet protocol. IETF RFC 760 (1980). http://www.ietf.org/rfc/rfc760.txt

[149] Price, G.: Web search engines FAQS: questions, answers, and issues. Searcher **9**(9) (2001). http://www.infotoday.comc/searcheroct01/price.htm

[150] Raghavan, S., Garcia-Molina, H.: Crawling the hidden web. In: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 129–138 (2001). http://dbpubs.stanford.edu/pub/2000-25

[151] Resource description framework. W3C Recommendation. http://www.w3.org/RDF/

[152] Reagle, J.: Web RSS (syndication) history (2003). http://goatee.net/2003/rss-history.html

[153] Robinson, G., Cargill, C.: History and impact of computer standards. Computer **29**(10), 79 – 85 (1996)

[154] Rosenthal, D., Robertson, T., Lipkis, T., Reich, T., Morabito, S.: Requirements for digital preservation systems. a bottom-up approach. D-Lib Magazine **11**(11) (2005). doi:10.1045/november2005-rosenthal

[155] The rosetta stone. Online photo by Eduardo Rocha, PhD. http://erocha.freehosting.net/Rosetta_Stone_WEB2.jpg

[156] Rosetta stone replica. From the Museum Store Company online catalog (2008). http://museumstorecompany.com/index.php?cPath=10

[157] Rothenberg, J.: Ensuring the longevity of digital information. Scientific American **272**(1), 42–47 (1995)

[158] Rothenberg, J.: Ensuring the longevity of digital information. http:www.clir.org/pubs/archives/ensuring.pdf (1999). Revision 980327

[159] S. Josefsson: The base16, base32, and base64 data encodings. IETF RFC 3548 (2003). http://www.ietf.org/rfc/rfc3548.txt

[160] Scott, M.: Wordsmith software package. Oxford University Press (2008). http://www.lexically.net/wordsmith/

[161] Sherman, C.: Getting the New York Times more search engine friendly. Search Engine Watch (2006). http://searchenginewatch.com/showPage.html?page=3613561

[162] Shirky, C.: Ontology is overrated: Categories, links, and tags. Written synthesis of two conference talks. http://www.shirky.com/writings/ontology_overrated.html

[163] Shirky, C.: AIHT Conceptual issues from practical tests. D-Lib Magazine **11**(12) (2005). http://www.dlib.org/dlib/december05/shirky/12shirky.html

[164] Shirky, C.: Library of Congress Archive Ingest and Handling Test AIHT final report. Report by the National Digital Information Infrastruction & Preservation Program (2005). http://www.digitalpreservation.gov/library/pdf/ndiipp_aiht_final_report.pdf

[165] Silverman, J.A.: Photographic evidence, naked children, and dead celebrities: Digital forgery and the law (1998). http://www.thirdamendment.com/digital.html

[166] Singh, S.: The Codebook: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Anchorbooks (2000)

[167] Sitemaps XML Format. Sitemaps Protocol (2008). http://www.sitemaps.org/protocol.php

[168] Smith, J., Nelson, M.: Using OAI-PMH resource harvesting and MPEG-21 DIDL for digital preservation. 2nd International Conference on Open Repositories 2007 (2007)

[169] Smith, J.A., Klein, M., Nelson, M.L.: Repository replication using NNTP and SMTP. In: Proceedings of the European Conference on Digital Libraries ECDL 2006, pp. 51–62 (2006)

[170] Smith, J.A., McCown, F., Nelson, M.L.: Observed web robot behavior on decaying web subsites. D-Lib Magazine (2006). http://www.dlib.org/dlib/february06/smith/02smith.html

[171] Smith, J.A., Nelson, M.L.: CRATE: A simple model for self-describing web resources. In: Proceedings of the 7th International Web Archiving Workshop IWAW'07 (2007)

[172] Smith, J.A., Nelson, M.L.: Generating best-effort preservation metadata for web resources at time of dissemination. In: Proceedings of the Joint Conference on Digital Libraries(JCDL 2007), pp. 51–52 (2007)

[173] Smith, J.A., Nelson, M.L.: Creating preservation-ready web resources. D-Lib Magazine **14**(1/2) (2008). doi:10.1045/january2008-smith

[174] Smith, J.A., Nelson, M.L.: A quantitative evaluation of dissemination-time preservation metadata. In: Proceedings of the European Conference on Digital Libraries ECDL 2008 (2008). (To appear)

[175] Smith, J.A., Nelson, M.L.: Site design impact on robots: An examination of search engine crawler behavior at deep and wide websites. D-Lib Magazine **14**(3/4) (2008). doi:10.1045/march2008-smith

[176] Suleman, H.: OAI-PMH2 XMLFile file-based data provider (2002). http://www.dlib.vt.edu/projects/OAI/software/xmlfile/xmlfile.html

[177] Sullivan, D.: A closer look at privacy & desktop search (2004). http://searchenginewatch.com/sereport/article.php/3421621

[178] Team, T.U.I.: Rosetta stone (1997). http://library.thinkquest.org/10005/media/photos/RosettaStone.gif

[179] Thomas, C.F., Griffin, L.S.: Who will Create The Metadata For the Internet? First Monday **3**(12) (1998). http://www.firstmonday.dk/issues/issue3_12/thomas/

[180] U.S. Department of Commerce: Federal Standard 1037C. telecommunications: Glossary of telecommunications terms (August). http://www.its.bldrdoc.gov/fs-1037

[181] Van de Sompel, H., Lagoze, C.: The Santa Fe Convention of the Open Archives Initiative. D-Lib Magazine **6**(2) (2000). http://www.dlib.org/dlib/february00/vandesompel-oai/02vandesompel-oai.html

[182] Van de Sompel, H., Lagoze, C.: Notes from the interoperability front: A progress report on the Open Archives Initiative. In: Proceedings of the European Conference on Digital Libraries ECDL'02, pp. 144–157 (2002)

[183] Van de Sompel, H., Nelson, M.L., Lagoze, C., Warner, S.: Resource harvesting within the OAI-PMH framework. D-Lib Magazine **10**(12) (2004). doi:10.1045/december2004-vandesompel

[184] Van de Sompel, H., Young, J.A., Hickey, T.B.: Using the OAI-PMH ... differently. D-Lib Magazine **9**(7/8) (2003). http://dx.doi.org/10.1045/july2003-young

[185] van Hoff, A., Giannandrea, J., Hapner, M., Carter, S., Medin, M.: The HTTP distribution and replication protocol (1997). http://www.w3.org/TR/NOTE-drp

[186] Veblen, T.: The Place of Science in Modern Civilization and Other Essays. B.W. Huebsch (1919)

[187] Victoria Electronic Records System: The step-by-step guide: A road map to a VERS implementation. http://www.prov.vic.gov.au/vers/toolkit/stepbystep/default.htm (2008)

[188] Web Characterization Terminology and Definition Sheet, W3C Working Draft 24-May-1999 (1999). http://www.w3.org/1999/05/WCA-terms/

[189] W3C: RDFa in XHTML: syntax and processing. Tech. rep., W3C (2008). http://www.w3.org/TR/2008/WD-rdfa-syntax-20080221/

[190] Walker, R.: The guts of a new machine. The New York Times (2003)

[191] WARP: National Diet Library of Japan, Web Archiving Project. (A summary report in English). http://www.ndl.go.jp/en/iflapac/e_resources.html

[192] Waugh, A.: The design of the VERS encapsulated object experience with an archival information package. International Journal on Digital Libraries **6**(2), 184–191 (2006)

[193] Waugh, A.: The design and implementation of an ingest function to a digital archive. D-Lib Magazine (2007). doi:10.1045/november2007-waugh

[194] Waugh, A., Wilkinson, R., Hills, B., Dell'oro, J.: Preserving digital information forever. In: DL '00: Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 175–184 (2000). http://doi.acm.org/10.1145/336597.336659

[195] The Wayback Machine Frequently Asked Questions. http://www.archive.org/about/faqs.php (Accessed 18 January 2006)

[196] Webb, C.: National Library of Australia preservation metadata for digital collections. http://www.nla.gov.au/preserve/pmeta.html (1999)

[197] Weibel, S.: Metadata: The foundations of resource description. D-Lib Magazine **1**(1) (1995). http://dx.doi.org/10.1045/july95-weibel

[198] Windows live search academic: Publishers frequently asked questions. http://academic.live.com/Publishers_Faq.htm (Accessed on July 10, 2006) (2006). http://academic.live.com/Publishers_Faq.htm

[199] XMLSITEMAP.COM: Xmlsitemap. Web-based sitemap tool. (2008). http://xmlsitemap.com/create-sitemap/

[200] Yahoo: Flickr: a subcategory of images with OAI-PMH tags (2006). http://flickr.com/photos/tags/oaipmh/

[201] Yergeau, F.: UTF-8, a transformation format of ISO 10646. IETF RFC 2279 (1998). http://www.ietf.org/rfc/rfc2279.txt

[202] Zimmerman, P.: The Official PGP User's Guide. MIT Press, Cambridge, MA (1995)

[203] Zimmerman, P.: Why I wrote PGP (1999). http://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html

APPENDIX A

SUPPLEMENTARY GRAPHS OF CRAWLER BEHAVIOR

The crawling patterns seen in the Deep-and-Wide website experiment differed for each of the major search engines, Google, Yahoo, and MSN. In addition, each crawler exhibited a different pattern at each of the sites, approaching .com and .edu sites at varying rates, for example and with varying coverage rates.

1 GOOGLE'S ROBOT (GOOGLEBOT)

Google's robot covered more of every site and crawled more often than MSN and Yahoo. Figure 69 shows the intermediate and final coverage by Google on the Buffet.com website, and Figure 70 graphs the coverage for the Bread-Crumb.com website.

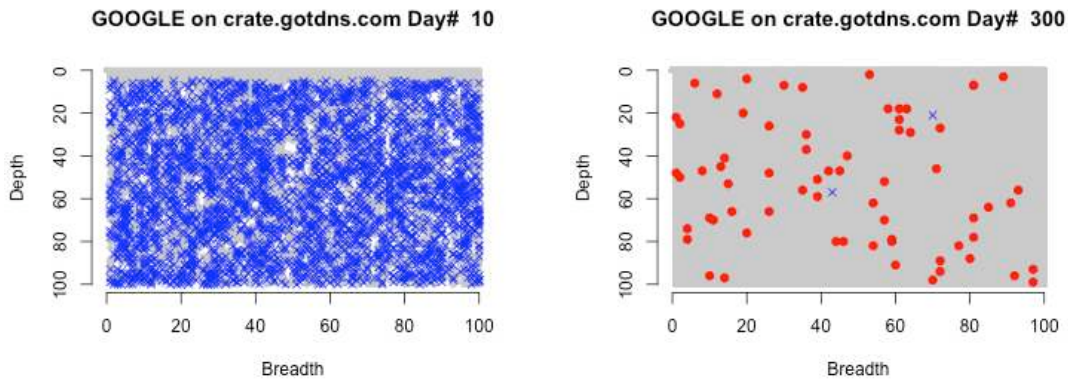


FIG. 69: Google on the Buffet.edu site. The left graph is a mid-point in the crawl of the site, and the graph on the right is the final coverage.

Contrast the Google Buffet.com and Bread-Crumb.com graphs with the Google Buffet.edu (Figure 71) and Bread-Crumb.edu graphs (in Figure 72). These show a much less aggressive crawling pattern, even though the only real difference between these .edu sites and the .com sites is the Top-Level-Domain where they are hosted. This implies that Google views these domains differently.

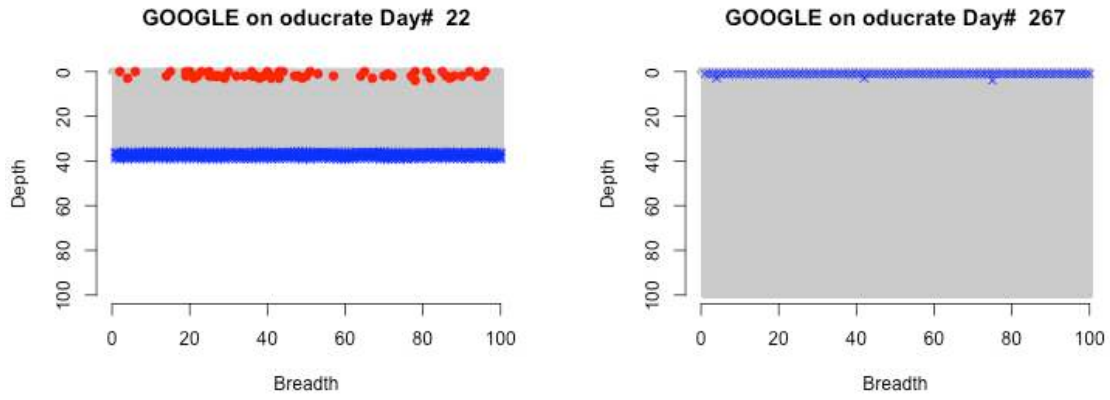


FIG. 70: Google on the Bread-Crumb.com site. The left graph is a mid-point in the crawl of the site, and the graph on the right is the final coverage.

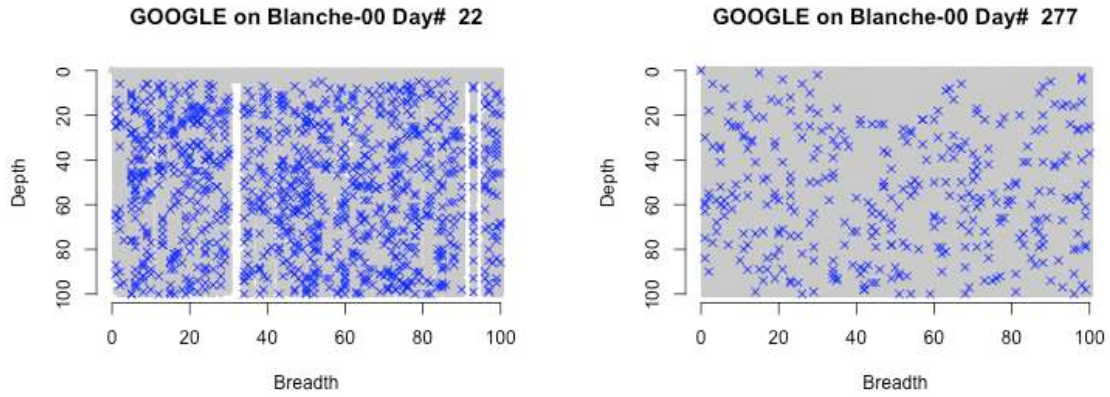


FIG. 71: Google on the Buffet.edu site. The left graph is at a mid-point in the crawl of the site, and the graph on the right is the final coverage day.

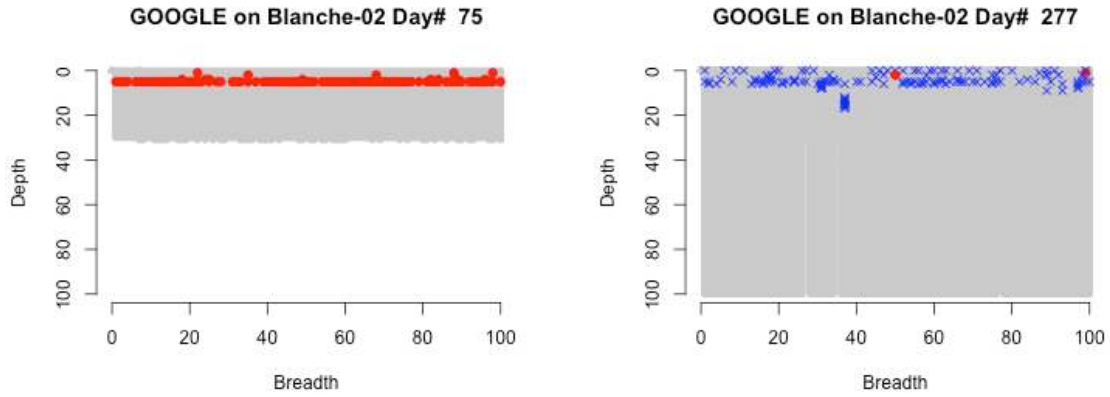


FIG. 72: Google on the Bread-Crumb.edu site. The left graph is at a mid-point in the crawl of the site, and the graph on the right is the final coverage day.

2 MSN'S ROBOT (MSNBOT)

MSN's robot was not as aggressive as Google's, but it did eventually completely crawl the Buffet.com site. Figure 73 shows the intermediate and final coverage by the MSN robot on the Buffet.com website. On the other hand, it explored very little of the Bread-Crumb.com website. The MSN robots show a clear preference for menu-style organization of a website.

Contrast the MSN robot Buffet.com and Bread-Crumb.com graphs with the MSN robot Buffet.edu (Figure 75) and Bread-Crumb.edu graphs (in Figure 76). MSN's robot eventually completed a crawl of the Buffet.edu website, but barely explored the Bread-Crumb.edu website. In this case, the crawler prejudice appears to be against the site design rather than the Top-Level Domain (versus Google's robot).

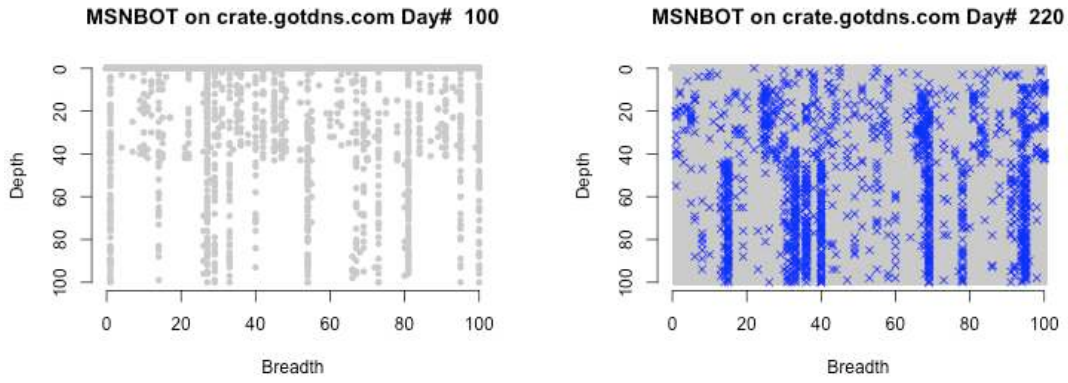


FIG. 73: MSN robot on the Buffet.com site. The left graph is a mid-point in the crawl of the site, and the graph on the right is the final coverage.

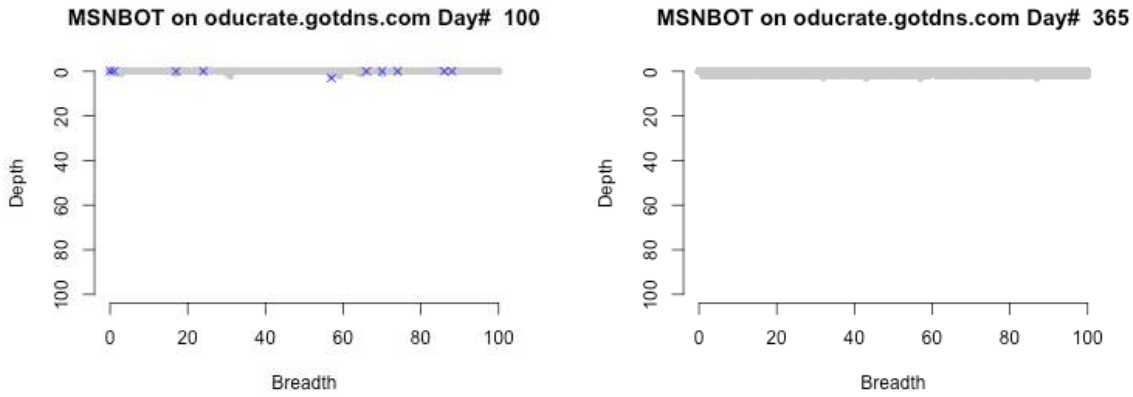


FIG. 74: MSN robot on the Bread-Crumb.com site. The left graph is a mid-point in the crawl of the site, and the graph on the right is the final coverage.

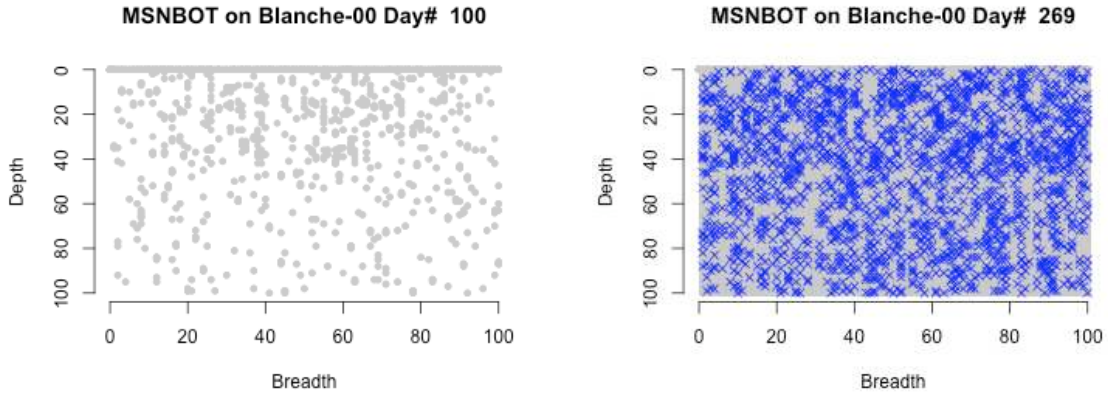


FIG. 75: MSN robot on the Buffet.edu site. The left graph is at a mid-point in the crawl of the site, and the graph on the right is the final coverage day.

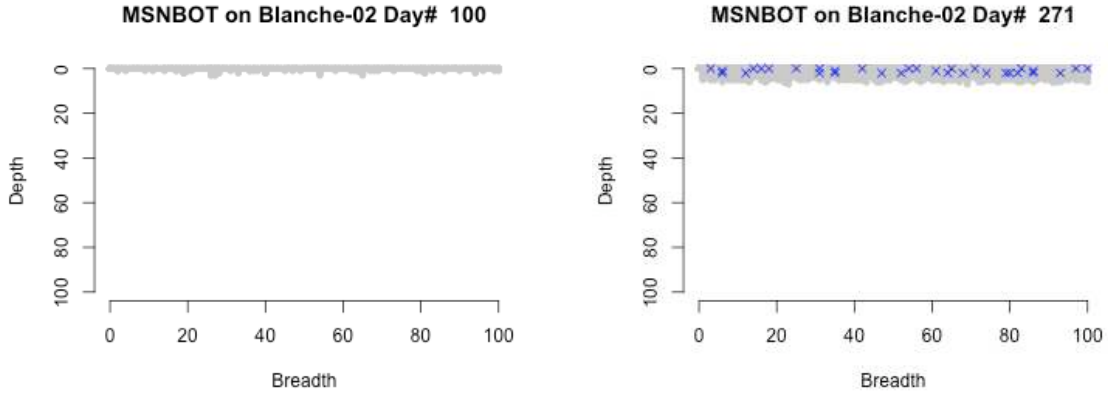


FIG. 76: MSN robot on the Bread-Crumb.edu site. The left graph is at a mid-point in the crawl of the site, and the graph on the right is the final coverage day.

3 YAHOO'S ROBOT (SLURP)

Yahoo's robot was not as aggressive as Google's or MSN's, and did very little exploring at any of the experiment's websites, regardless of link organization. Its highest coverage rate, at the Buffet.com site, was still merely 44% even though it had generated 50,378 requests (cf. Table 12). Figure 77 shows the intermediate and final coverage by the Yahoo robot on the Buffet.com website.

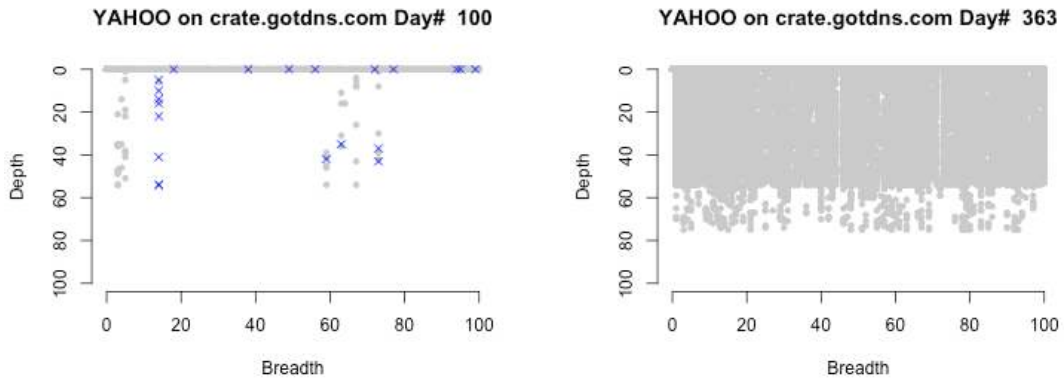


FIG. 77: Yahoo robot on the Buffet.com site. The left graph is a mid-point in the crawl of the site, and the graph on the right is the final coverage.

Yahoo also exhibited a preference for a URL-menu style of website design. It crawled less than 5% of the Bread-Crumb style sites, regardless of Top-Level Domain.

Contrast the Yahoo robot crawls on the Buffet.com and Bread-Crumb.com graphs with the Yahoo robot crawls on the Buffet.edu (Figure 79) and Bread-Crumb.edu graphs (in Figure 80). To speak anthropomorphically, the Yahoo robot seems uninterested in the Bread-Crumb.edu website, and only marginally interested in the Buffet.edu website.

4 SUMMARY

These graphs support the conventional wisdom that site design matters to search engine robots. Sites that specialize in providing advice to website designers[1] recommend that designers provide a top-level index to the website's URLs, and to make websites wide rather than deep. While Google apparently cares less about this aspect of website design, the other search engines do exhibit such a preference. Website owners who want to maximize coverage across all search engines would likely be well served by following this advice.

[1] For example, `http://searchenginewatch.com/` and `http://www.seo.com/`

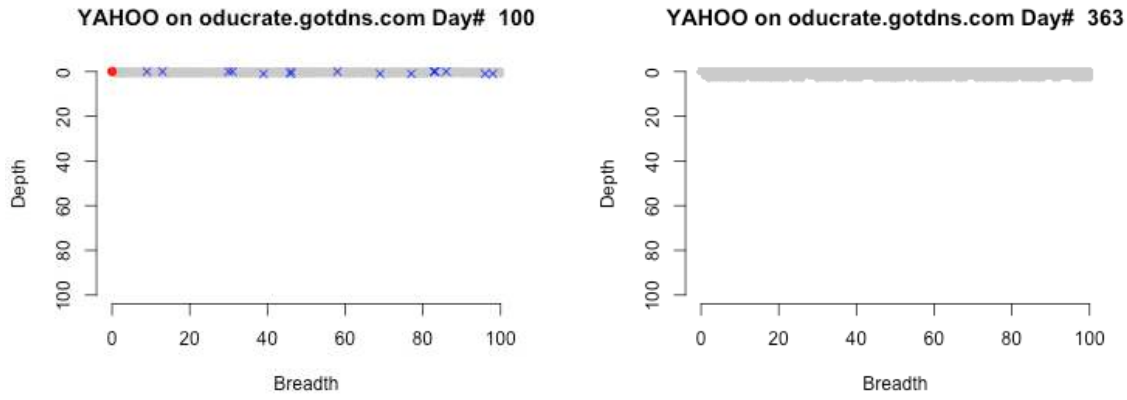


FIG. 78: Yahoo robot on the Bread-Crumb.com site. The left graph is a mid-point in the crawl of the site, and the graph on the right is the final coverage.

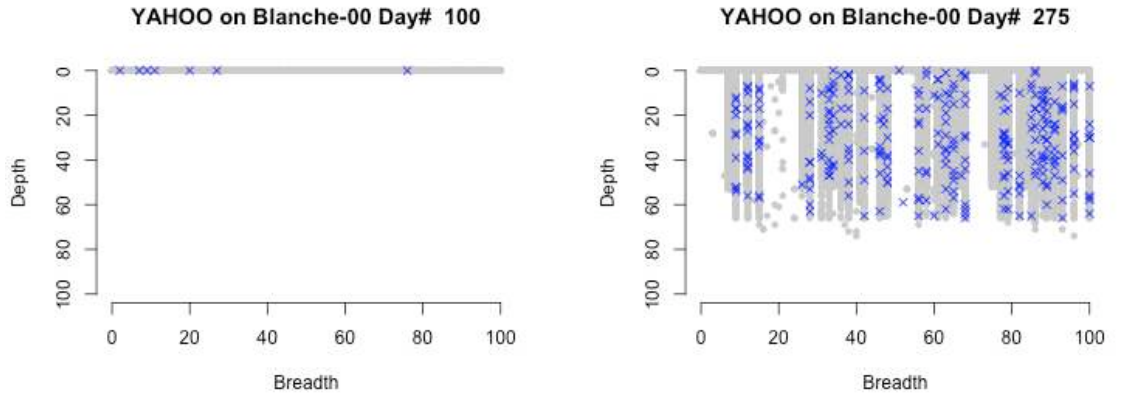


FIG. 79: Yahoo robot on the Buffet.edu site. The left graph is at a mid-point in the crawl of the site, and the graph on the right is the final coverage day.

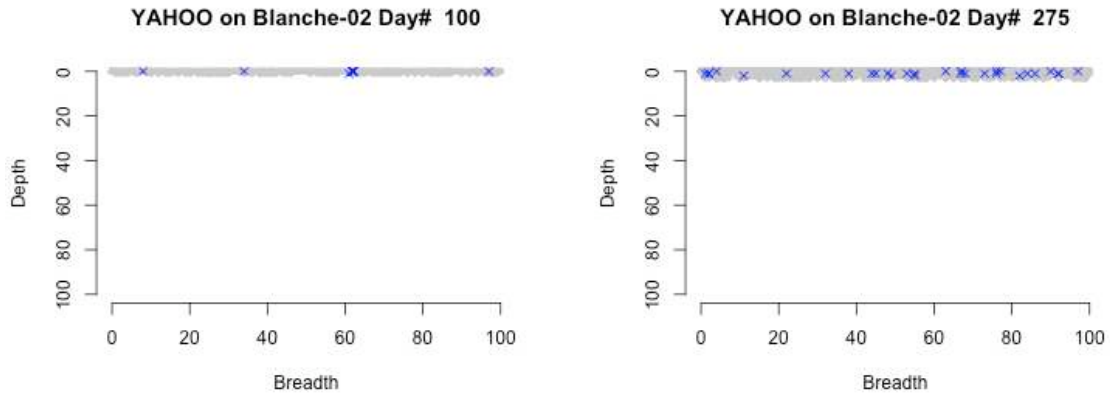


FIG. 80: Yahoo robot on the Bread-Crumb.edu site. The left graph is at a mid-point in the crawl of the site, and the graph on the right is the final coverage day.

APPENDIX B

CRATE XML SCHEMA DOCUMENTS

1 THE SIMPLE CRATE SCHEMA: CRATE.XSD

```
<?xml version="1.0"?>
<xsd:annotation>
<xsd:documentation>
   This is a schema to represent the CRATE model.
   Schema author: Joan A. Smith
</xsd:documentation>
</xsd:annotation>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.cratemodel.org/xmlns/"
   elementFormDefault="qualified">
<xsd:element name="Crate" minOccurs="1" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="ResourceID" type="xsd:string"/>
<xsd:element name="MetadataUtility"
type="MetadataUtilityType"
   minOccurs="1" maxOccurs="unbounded"/>
<xsd:element name="Resource"
   type="ResourceType" minOccurs="1">
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:complexType name="MetadataUtilityType">
<xsd:sequence>
<xsd:element name="Name" type="xsd:string"
   minOccurs="1" maxOccurs="1"/>
<xsd:element name="Exec" type="xsd:string"
   minOccurs="0" maxOccurs="1"/>
<xsd:element name="Version" type="xsd:string"
   minOccurs="1" maxOccurs="1"/>
<xsd:element name="MimeSet" type="xsd:string"
   minOccurs="0" maxOccurs="1"/>
<xsd:element name="ExecTimeStamp" type="xsd:timeInstant"
   minOccurs="0"/>
<xsd:element name="ResourceMetadata" type="xsd:string"
   minOccurs="0" maxOccurs="1"/>
<xsd:any namespace="http://www.cratemodel.org/xmlns/"
   minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ResourceType">
<xsd:sequence>
<xsd:element name="Base64Encoded" minOccurs="0"/>
<xsd:element name="ByReferenceURI" minOccurs="0"/>
```

```
<xsd:element name="Other" minOccurs="0">
<xsd:complexType>
<xsd:sequence>
<any minOccurs="0"
   maxOccurs="unbounded"
   processContents="lax"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:schema>
```

2 CRATE LANL DIDL SCHEMA: OAICRATE.XSD

```
<?xml version="1.0"?>
<xsd:annotation>
<xsd:documentation>
   This is a schema to extend the oai_didl
   content created by mod_oai with CRATE
   model plugin information.
   Schema author: Joan A. Smith,
      Old Dominion University
</xsd:documentation>
</xsd:annotation>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.modoai.org/xmlns/"
   elementFormDefault="qualified">
<xsd:element name="CrateUtilities"
   minOccurs="1" maxOccurs="unbounded">
<xsd:complexType>
<xsd:element name="Plugin" type="PluginType"
   minOccurs="1" maxOccurs="unbounded"/>
</xsd:complexType>
</xsd:element>
<xsd:complexType name="PluginType">
<xsd:sequence>
<xsd:element name="Name" type="xsd:string"
   minOccurs="1" maxOccurs="1"/>
<xsd:element name="Exec" type="xsd:string"
   minOccurs="0" maxOccurs="1"/>
<xsd:element name="Version" type="xsd:string"
   minOccurs="1" maxOccurs="1"/>
<xsd:element name="MimeSet" type="xsd:string"
   minOccurs="0" maxOccurs="1"/>
<xsd:any namespace="http://www.modoai.org/xmlns/"
   minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

APPENDIX C

EXAMPLE CRATE RESPONSES

1 CRATE PLUGINS IN THE IDENTIFY RESPONSE

When MODOAI has been installed together with metadata utility plugins, the plugins are listed in the response to the OAI-PMH Identify verb (line numbers 54 –174). The plugin description also specifies the rules, i.e., which resources will be processed by the plugin. Compare line 62 (applies to any text-based resource) and 166 (applies only to Zip-type files) in this section, for example, with line 83 which applies to every resource.

REQUEST: `http://www.foo.edu:8080/modoai?verb=Identify`

RESPONSE:

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
  http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
<responseDate>2008-05-15T00:04:08Z</responseDate>
<request verb="Identify">
  http://www.foo.edu:8080/modoai</request>
<Identify>
<repositoryName>http://www.foo.edu:8080</repositoryName>
<baseURL>http://www.foo.edu:8080/modoai</baseURL>
<protocolVersion>2.0</protocolVersion>
<adminEmail>jsmit@cs.odu.edu</adminEmail>
<earliestDatestamp>1900-01-01T12:00:00Z</earliestDatestamp>
<deletedRecord>no</deletedRecord>
<granularity>YYYY-MM-DDThh:mm:ssZ</granularity>
<description>
<friends
  xmlns="http://www.openarchives.org/OAI/2.0/friends/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.openarchives.org/OAI/2.0/friends/
  http://www.openarchives.org/OAI/2.0/friends.xsd">
</friends>
</description>
<description>
<gateway
  xmlns="http://www.openarchives.org/OAI/2.0/gateway/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xsi:schemaLocation=
     "http://www.openarchives.org/OAI/2.0/gateway/
      http://www.openarchives.org/OAI/2.0/gateway.xsd">
  <source>http://www.foo.edu:8080</source>
  <gatewayDescription>
    http://www.modoai.org/gateway.html
  </gatewayDescription>
  <gatewayAdmin>mail@joanasmith.com</gatewayAdmin>
  </gateway>
  </description>
  <description>
  <modoai
    xmlns:modoai="http://www.modoai.org/OAI/2.0/modoai/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
      "http://www.modoai.org/OAI/2.0/modoai/modoai.xsd">
  <modoai:version>0.7.1</modoai:version>
  <modoai:server>
    Apache/2.2.4 (Ubuntu)
    PHP/5.2.3-1ubuntu6.3
  </modoai:server>
  </modoai>
  </description>
  <description>
  <cratePlugins
    xmlns="http://cratemodel.org/OAI/2.0/cratePlugins/"
    xmlns:crateplugin="http://www.modoai.org/OAI/2.0/modoai/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
      "http://cratemodel.org/OAI/2.0/cratePlugins/
       http://cratemodel.org/OAI/2.0/cratePlugins.xsd">
  <crateplugin>
  <crateplugin:name>wc</crateplugin:name>
  <crateplugin:exec>/usr/bin/wc</crateplugin:exec>
  <crateplugin:version>
  <![CDATA[wc (GNU coreutils) 5.97
    Copyright (C) 2006 Free Software Foundation, Inc.
    This is free software.
    You may redistribute copies of it under the terms of
    the GNU General Public License
    <http://www.gnu.org/licenses/gpl.html>.
    There is NO WARRANTY, to the extent permitted by law.
    Written by Paul Rubin and David MacKenzie.]]>
  </crateplugin:version>
  <crateplugin:mimeSet>text/*</crateplugin:mimeSet>
  </crateplugin>
  <crateplugin>
  <crateplugin:name>file</crateplugin:name>
  <crateplugin:exec>/usr/bin/file</crateplugin:exec>
  <crateplugin:version>
  <![CDATA[file-4.21 magic file from
    /etc/magic:/usr/share/file/magic]]>
  </crateplugin:version>
```

```
<crateplugin:mimeSet>*/*</crateplugin:mimeSet>
</crateplugin>
<crateplugin>
<crateplugin:name>md5sum</crateplugin:name>
<crateplugin:exec>/usr/bin/md5sum</crateplugin:exec>
<crateplugin:version>
<![CDATA[md5sum (GNU coreutils) 5.97
  Copyright (C) 2006 Free Software Foundation, Inc.
  This is free software.
  You may redistribute copies of it under the terms of
  the GNU General Public License
  <http://www.gnu.org/licenses/gpl.html>.
  There is NO WARRANTY, to the extent permitted by law.
  Written by Ulrich Drepper, Scott Miller,
  and David Madore.]]>
</crateplugin:version>
<crateplugin:mimeSet>application/*</crateplugin:mimeSet>
</crateplugin>
<crateplugin>
<crateplugin:name>sha1sum</crateplugin:name>
<crateplugin:exec>/usr/bin/sha1sum</crateplugin:exec>
<crateplugin:version>
<![CDATA[sha1sum (GNU coreutils) 5.97
  Copyright (C) 2006 Free Software Foundation, Inc.
  This is free software.
  You may redistribute copies of it under the terms of
  the GNU General Public License
  <http://www.gnu.org/licenses/gpl.html>.
  There is NO WARRANTY, to the extent permitted by law.
  Written by Ulrich Drepper, Scott Miller,
  and David Madore.]]>
</crateplugin:version>
<crateplugin:mimeSet>image/*</crateplugin:mimeSet>
</crateplugin>
<crateplugin>
<crateplugin:name>sha224sum</crateplugin:name>
<crateplugin:exec>/usr/bin/sha224sum</crateplugin:exec>
<crateplugin:version>
<![CDATA[sha224sum (GNU coreutils) 5.97
  Copyright (C) 2006 Free Software Foundation, Inc.
  This is free software.
  You may redistribute copies of it under the terms of
  the GNU General Public License
  <http://www.gnu.org/licenses/gpl.html>.
  There is NO WARRANTY, to the extent permitted by law.
  Written by Ulrich Drepper, Scott Miller,
  and David Madore.]]>
</crateplugin:version>
<crateplugin:mimeSet>image/png</crateplugin:mimeSet>
</crateplugin>
<crateplugin>
<crateplugin:name>sha384sum</crateplugin:name>
<crateplugin:exec>/usr/bin/sha384sum</crateplugin:exec>
```

```
<crateplugin:version>
<![CDATA[sha384sum (GNU coreutils) 5.97
  Copyright (C) 2006 Free Software Foundation, Inc.
  This is free software.
  You may redistribute copies of it under the terms of
  the GNU General Public License
  <http://www.gnu.org/licenses/gpl.html>.
  There is NO WARRANTY, to the extent permitted by law.
  Written by Ulrich Drepper, Scott Miller,
  and David Madore.]]>
</crateplugin:version>
<crateplugin:mimeSet>image/jpeg</crateplugin:mimeSet>
</crateplugin>
<crateplugin>
<crateplugin:name>sha256sum</crateplugin:name>
<crateplugin:exec>/usr/bin/sha256sum</crateplugin:exec>
<crateplugin:version>
<![CDATA[sha256sum (GNU coreutils) 5.97
  Copyright (C) 2006 Free Software Foundation, Inc.
  This is free software.
  You may redistribute copies of it under the terms of
  the GNU General Public License
  <http://www.gnu.org/licenses/gpl.html>.
  There is NO WARRANTY, to the extent permitted by law.
  Written by Ulrich Drepper, Scott Miller,
  and David Madore.]]>
</crateplugin:version>
<crateplugin:mimeSet>application/pdf</crateplugin:mimeSet>
</crateplugin>
<crateplugin>
<crateplugin:name>zipinfo</crateplugin:name>
<crateplugin:exec>/usr/bin/zipinfo</crateplugin:exec>
<crateplugin:version>
<![CDATA[ZipInfo 2.42 of 28 February 2005,
  by Greg Roelofs and the Info-ZIP group.]]>
</crateplugin:version>
<crateplugin:mimeSet>*/zip</crateplugin:mimeSet>
</crateplugin>
</cratePlugins>
</description>
</Identify>
</OAI-PMH>
```

2 CRATE PLUGINS IN THE GET RECORD RESPONSE

Plugins are applied to each resource based on criteria specified in the configuration file, *modoai.conf*. In the example below, the *word count* (wc) utility is not applied because the resource is a JPEG image. Cf. lines 46 – 59 in Appendix C–1 where the rules for that plugin are defined.

REQUEST:

```
http://www.foo.edu:8080/modoai?verb=GetRecord
    &metadataPrefix=oai_crate
    &identifier=http://www.foo.edu/modoaitest/crate.jpeg
```

RESPONSE:

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
<responseDate>2008-05-15T15:32:50Z</responseDate>
<request verb="GetRecord"
identifier="http://www.foo.edu/modoaitest/crate.jpeg"
metadataPrefix="oai_crate">http://www.foo.edu:8080/modoai/</request>
<GetRecord>
<record>
<header>
<identifier>http://www.foo.edu/modoaitest/crate.jpeg</identifier>
<datestamp>2000-02-28T17:00:00Z</datestamp>
<setSpec>mime:image:jpeg</setSpec>
</header>
<metadata>
<didl:DIDL xmlns:didl="urn:mpeg:mpeg21:2002:02-DIDL-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
http://purl.lanl.gov/STB-RL/schemas/2004-11/DIDL.xsd">
<didl:Item>
<didl:Descriptor>
<didl:Statement mimeType="application/xml; charset=utf-8">
<dii:Identifier xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2002:01-DII-NS
http://purl.lanl.gov/STB-RL/schemas/2003-09/DII.xsd">
http://www.foo.edu/modoaitest/crate.jpeg</dii:Identifier>
</didl:Statement>
</didl:Descriptor>
<didl:Descriptor>
<didl:Statement mimeType="application/xml; charset=utf-8">
<http:header xmlns:http="http://www.modoai.org/OAI/2.0/http_header/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.modoai.org/OAI/2.0/http_header/
http://purl.lanl.gov/STB-RL/schemas/2004-08/HTTP-HEADER.xsd">
<http:Content-Length>19339</http:Content-Length>
<http:Server>Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6.3</http:Server>
```

```
<http:Content-Type>image/jpeg</http:Content-Type>
<http:Last-Modified>Mon, 28 Feb 2000 17:00:00 GMT</http:Last-Modified>
<http:Date>Thu, 15 May 2008 15:32:50 GMT</http:Date>
<http:Via>1.1 SRVWINISA003</http:Via><http:User-Agent>Mozilla/4.0
(compatible; MSIE 5.5; Windows 98)</http:User-Agent>
<http:Host>www.foo.edu:8080</http:Host>
<http:Te>deflate,gzip;q=0.3</http:Te>
<http:X-Forwarded-For>70.161.101.174</http:X-Forwarded-For>
<http:Accept>image/gif, image/x-xbitmap,
image/jpeg, image/pjpeg, application/vnd.ms-excel,
application/msword, application/vnd.ms-powerpoint,
*/*</http:Accept>
<http:Connection>Keep-Alive</http:Connection></http:header>
</didl:Statement>
</didl:Descriptor>
<oai_crate:crateplugin
xmlns:oai_crate='http://modoai.org/OAI/2.0/oai_crate/'
xmlns:crateplugin="http://modoai.org/OAI/2.0/crateplugin "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance "
xsi:schemaLocation="http://modoai.org/OAI/2.0/oai_crate/crate.xsi" >
<crateplugin>
<crateplugin:name>file</crateplugin:name>
<crateplugin:version>
<![CDATA[file-4.21 magic file from /etc/magic:/usr/share/file/magic
]]>
</crateplugin:version>
<crateplugin:content>
<![CDATA[
/var/www/modoaitest/crate.jpeg:
JPEG image data, JFIF standard 1.01
]]>
</crateplugin:content>
</crateplugin>
<crateplugin>
<crateplugin:name>sha1sum</crateplugin:name>
<crateplugin:version>
<![CDATA[sha1sum (GNU coreutils) 5.97
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software.  You may redistribute copies of it under the
terms of the GNU General Public License
<http://www.gnu.org/licenses/gpl.html>.
There is NO WARRANTY, to the extent permitted by law.
Written by Ulrich Drepper, Scott Miller, and David Madore.
]]>
</crateplugin:version>
<crateplugin:content>
<![CDATA[7b15663fbfb3bc174c5883d2b57facbe91465bcb
/var/www/modoaitest/crate.jpeg
]]>
</crateplugin:content>
</crateplugin>
<crateplugin>
<crateplugin:name>sha384sum</crateplugin:name>
```

```
<crateplugin:version>
<![CDATA[sha384sum (GNU coreutils) 5.97
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software.  You may redistribute copies of it under the
terms of the GNU General Public License
<http://www.gnu.org/licenses/gpl.html>.
There is NO WARRANTY, to the extent permitted by law.
Written by Ulrich Drepper, Scott Miller, and David Madore.
]]>
</crateplugin:version>
<crateplugin:content>
<![CDATA[709f27aa6832e044fb7edaab2abd0fec00cb478dddeb68674
4369c6bbd1bbb1aa2052ecdf58a73b1945dd69150583bb6
/var/www/modoaitest/crate.jpeg
]]>
</crateplugin:content>
</crateplugin>
</oai_crate:crateplugin>
<didl:Component>
<didl:Resource mimeType="image/jpeg"
encoding="base64">/9j/4AAQSkZJRgBDAAE.....IWzr9e3+1P/2Q==
</didl:Resource>
<didl:Resource mimeType="image/jpeg"
ref="http://www.foo.edu/modoaitest/crate.jpeg"/>
</didl:Component>
</didl:Item>
</didl:DIDL>
</metadata>
</record>
</GetRecord>
</OAI-PMH>
```

3 OAI-PMH LIST IDENTIFIERS RESPONSE

The presence of CRATE plugins in MODOAI does not affect the response to an OAI-PMH List Identifiers request. This response essentially rewrites the contents of the sitemap file in OAI-PMH XML form rather than in the sitemap protocol format. Cf. Appendix F for more details on sitemap files.

REQUEST:

```
http://www.foo.edu:8080/modoai?verb=ListIdentifiers
```

RESPONSE:

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
<responseDate>2008-05-15T15:31:10Z</responseDate>
<request verb="ListIdentifiers" metadataPrefix="oai_crate">
http://www.foo.edu:8080/modoai/</request>
<ListIdentifiers>
<header>
<identifier>http://www.foo.edu:8080/modoaitest/NetBeansAntTasks.zip
</identifier>
<datestamp>2007-09-19T08:06:56Z</datestamp>
<setSpec>mime:application:zip</setSpec>
</header>
<header>
<identifier>http://www.foo.edu:8080/modoaitest/README.txt</identifier>
<datestamp>2008-04-26T16:37:57Z</datestamp>
<setSpec>mime:text:plain</setSpec>
</header>
<header>
<identifier>http://www.foo.edu:8080/modoaitest/crate.jpeg</identifier>
<datestamp>2000-02-28T17:00:00Z</datestamp>
<setSpec>mime:image:jpeg</setSpec>
</header>
<header>
<identifier>http://www.foo.edu:8080/modoaitest/diag.jpg</identifier>
<datestamp>2002-10-22T16:00:00Z</datestamp>
<setSpec>mime:image:jpeg</setSpec>
</header>
<header>
<identifier>http://www.foo.edu:8080/modoaitest/file.sxw</identifier>
<datestamp>2005-12-31T17:00:00Z</datestamp>
<setSpec>mime:application:vnd.sun.xml.writer</setSpec>
</header>
<resumptionToken>5*oai_crate*0*0*0</resumptionToken></ListIdentifiers>
</OAI-PMH>
```

APPENDIX D

EXAMPLES OF METADATA UTILITY OUTPUT

1 COMPARATIVE METADATA OUTPUT OF A SMALL JPEG FILE

In the following sections, a small JPEG file is passed to utilities that were used as part of the CRATE evaluation experiments. The image is shown in Figure 81. The utilities used are Pronom-DROID, Jhove, Exif Tool, and the Unix-based utility, "File Magic". For additional comparison, we present the metadata that an HTTP Response produces (HTTP Headers), and the information available via the graphical KDE Desktop.

1.1 HTTP-Headers

Only a very simple set of headers is returned when requesting this file from the web server:

```
The HTTP Request:

HEAD /foo2.jpg HTTP/1.1
Host: localhost

The HTTP Response Headers:

HTTP/1.1 200 OK
Date: Wed, 14 May 2008 20:47:51 GMT
Server: Apache/2.2.8 (Ubuntu)
Last-Modified: Wed, 14 May 2008 20:35:46 GMT
ETag: "22e10d-6aac-44d36b9b40c80"
Accept-Ranges: bytes
Content-Length: 27308
Content-Type: image/jpeg
```

1.2 The Linux *file* Utility

The Unix/Linux File Magic utilities are closely associated with MIME typing as used by Apache and other web servers. As such, the bare minimum information is obtained from the file, as this example shows.

```
file-4.21
magic file from /etc/magic:/usr/share/file/magic
images/foo2.jpg: JPEG image data, JFIF standard 1.01 file-4.21
```

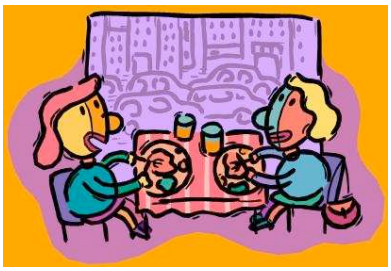


FIG. 81: .
“Two foos having lunch”. This image is from the foo2.jpg file. Appendix D–1 shows the metadata provided from the HTTP response, the Linux file utility, Pronom-DROID, ExifTool, and Jhove, all of which are command-line utilities; and by the KDE Desktop file inspector which provides a GUI-based view of file metadata.

1.3 Pronom-DROID

The Pronom-DROID utility provides somewhat more information than we have gleaned so far.

```
<?xml version="1.0" encoding="UTF-8"?>
<FileCollection
xmlns="http://www.nationalarchives.gov.uk/pronom/FileCollection">
<DROIDVersion>V1.1</DROIDVersion>
<SignatureFileVersion>12</SignatureFileVersion>
<DateCreated>2007-09-06T23:11:12</DateCreated>
<IdentificationFile IdentQuality="Positive" >
<FilePath>images/foo2.jpg</FilePath>
<FileFormatHit>
<Status>Positive (Specific Format)</Status>
<Name>JPEG File Interchange Format</Name>
<Version>1.01</Version>
<PUID>fmt/43</PUID>
</FileFormatHit>
</IdentificationFile>
</FileCollection>
```

1.4 ExifTool

Generally, Exif Tool was written to analyze digital camera photographs. While our sample Figure 81 is not a product of a digital camera, it is in JPEG format and so Exif provides some metadata, though much less than would be seen if it were a true digital camera image.

```
ExifTool Version Number         : 6.95
File Name                       : foo2.jpg
Directory                       : /home/jsmit/images
File Size                       : 27 kB
File Modification Date/Time     : 2007:08:31 13:02:25
File Type                       : JPEG
MIME Type                       : image/jpeg
JFIF Version                    : 1.1
Resolution Unit                 : inches
X Resolution                    : 150
Y Resolution                    : 150
Image Width                     : 409
Image Height                    : 278
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 409x278
```

1.5 Jhove with the JPEG-hul

Jhove offers a variety of filters which target specific file types, such as the eponymous JPEG-hul. Here is what Jhove reports for foo2.jpg:

```
Jhove (Rel. 1.1, 2006-06-05)
Date: 2008-05-14 13:30:00 EDT
RepresentationInformation: ../foo2.jpg
ReportingModule: JPEG-hul, Rel. 1.2 (2005-08-22)
LastModified: 2008-05-14 10:51:25 EDT
Size: 27308
Format: JPEG
Version: 1.01
Status: Well-Formed and valid
SignatureMatches:
JPEG-hul
MIMEtype: image/jpeg
Profile: JFIF
JPEGMetadata:
CompressionType: Huffman coding, Baseline DCT
Images:
Number: 1
Image:
NisoImageMetadata:
MIMEType: image/jpeg
ByteOrder: big-endian
CompressionScheme: JPEG
ColorSpace: YCbCr
SamplingFrequencyUnit: inch
XSamplingFrequency: 150
YSamplingFrequency: 150
ImageWidth: 408
ImageLength: 278
BitsPerSample: 8, 8, 8
SamplesPerPixel: 3
Scans: 1
QuantizationTables:
QuantizationTable:
Precision: 8-bit
DestinationIdentifier: 0
QuantizationTable:
Precision: 8-bit
DestinationIdentifier: 1
ApplicationSegments: APP0
```

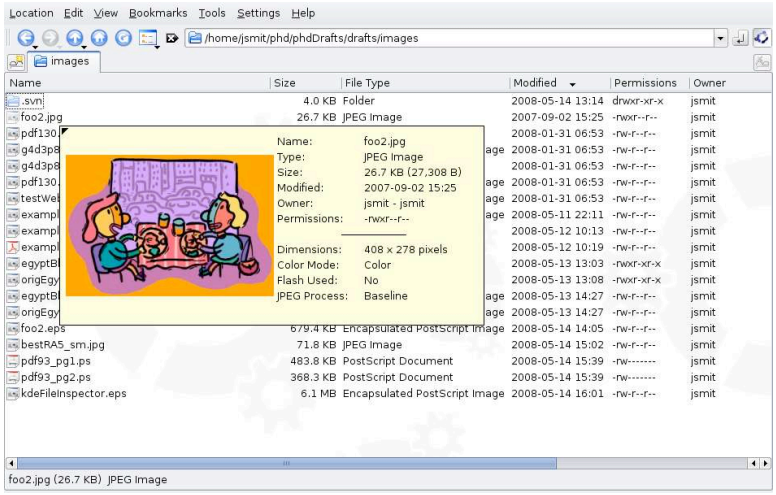


FIG. 82: .
The sample image file information seen from the KDE Desktop environment. The mouse cursor is placed on the file icon, and the information window appears beside it.

1.6 KDE Desktop File Inspector

By comparison, the KDE file inspector reports a bit more information about the file. This information is provided simply by moving the mouse pointer over the file icon or filename, as shown in Figure 82.

```
Name:          foo2.jpg
Type:          JPEG Image
Size:          26.7 KB (27,308 B)
Modified:      2007-09-02 15:25
Owner:         jsmit - jsmit
Permissions:   -rwxr--r--
Dimensions:    108 x 278 pixels
Color Mode:    Color
Flash Used:    No
JPEG Process:  Baseline
```

FIG. 83: Sample digital photograph analyzed with Exif Tool. A real digital photograph, taken with a Panasonic Lumix camera. Today's digital cameras often store a considerable amount of metadata with each image, but special tools are needed to access it. (L-R: Dr. Michael Overstreet, the author, Dr. Michael Nelson)

2 OTHER EXAMPLES OF METADATA UTILITY OUTPUT

2.1 Exif Tool Applied to a Digital Photograph

Exif Tool was designed specifically for digital photographs. Its output is much more detailed when applied to such a file rather than to the non-photographic image of Figure 81 as shown in Appendix D. The following analysis is produced by Exif Tool applied to the photograph shown in Figure 83.

EXIF Tool Output

```
bestRA5.JPG:
Intel format
 IFD 0 (Image) at offset 8:
   Make: (0x010F) ASCII=Panasonic @ 146
   Model: (0x0110) ASCII=DMC-TZ3 @ 156
   Orientation: (0x0112) Short=1 @ 42
   XResolution: (0x011A) Ratio=72 @ 164
   YResolution: (0x011B) Ratio=72 @ 172
   ResolutionUnit: (0x0128) Short=Pixels/Inch @ 78
   Software: (0x0131) ASCII=Ver.1.0   @ 180
   DateTime: (0x0132) ASCII=2008:05:03 11:56:44 @ 190
   YCbCrPositioning: (0x0213) Short=2 @ 114
   ExifOffset: (0x8769) Long=418 @ 126
   Tag 0xC4A5: (0xC4A5) Undefined=[80, 114, 105, 110, 116,
   73, 77, 0, 48, 50, 53, 48, 0, 0, 14, 0, 1, 0, 22, 0, 22,
   0, 2, 0, 0, 0, 0, 0, 3, 0, 100, 0, 0, 0, 7, 0, 0, 0, 0,
   0, 8, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0,
   11, 0, 172, 0, 0, 0, 12, 0, 0, 0, 0, 0, 13, 0, 0, 0, 0,
   0, 14, 0, 196, 0, 0, 0, 0, 1, 5, 0, 0, 0, 1, 1, 1, 0, 0,
```

```
  0, 16, 1, 128, 0, 0, 0, 9, 17, 0, 0, 16, 39, 0, 0, 11, 15,
  0, 0, 16, 39, 0, 0, 151, 5, 0, 0, 16, 39, 0, 0, 176, 8, 0,
  0, 16, 39, 0, 0, 1, 28, 0, 0, 16, 39, 0, 0, 94, 2, 0, 0,
  16, 39, 0, 0, 139, 0, 0, 0, 16, 39, 0, 0, 203, 3, 0, 0, 16,
  39, 0, 0, 229, 27, 0, 0, 16, 39, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0] @ 210
EXIF SubIFD at offset 418:
  ExposureTime: (0x829A) Ratio=10/2500 @ 832
  FNumber: (0x829D) Ratio=33/10 @ 840
  ExposureProgram: (0x8822) Short=Program Normal @ 452
  ISOSpeedRatings: (0x8827) Short=100 @ 464
  ExifVersion: (0x9000) Undefined=[48, 50, 50, 49] @ 476
  DateTimeOriginal: (0x9003) ASCII=2008:05:03 11:56:44 @ 848
  DateTimeDigitized: (0x9004) ASCII=2008:05:03 11:56:44 @ 868
  ComponentsConfiguration: (0x9101) Undefined=YCbCr @ 512
  CompressedBitsPerPixel: (0x9102) Ratio=4 @ 888
  ExposureBiasValue: (0x9204) Signed Ratio=0/100 @ 896
  MaxApertureValue: (0x9205) Ratio=344/100 @ 904
  MeteringMode: (0x9207) Short=5 @ 560
  LightSource: (0x9208) Short=Unknown @ 572
  Flash: (0x9209) Short=Auto Off @ 584
  FocalLength: (0x920A) Ratio=46/10 @ 912
  MakerNote: (0x927C) Undefined=[] @ 920
  FlashPixVersion: (0xA000) Undefined=[48, 49, 48, 48] @ 620
  ColorSpace: (0xA001) Short=1 @ 632
  ExifImageWidth: (0xA002) Long=3328 @ 644
  ExifImageLength: (0xA003) Long=1872 @ 656
  InteroperabilityOffset: (0xA005) Long=7702 @ 668
  SensingMethod: (0xA217) Short=2 @ 680
  FileSource: (0xA300) Undefined=Digital Camera @ 692
  SceneType: (0xA301) Undefined=Directly Photographed @ 704
  Tag 0xA401: (0xA401) Short=0 @ 716
  Tag 0xA402: (0xA402) Short=0 @ 728
  Tag 0xA403: (0xA403) Short=0 @ 740
  Tag 0xA404: (0xA404) Ratio=0/10 @ 7694
  Tag 0xA405: (0xA405) Short=28 @ 764
  Tag 0xA406: (0xA406) Short=0 @ 776
  Tag 0xA407: (0xA407) Short=0 @ 788
  Tag 0xA408: (0xA408) Short=0 @ 800
  Tag 0xA409: (0xA409) Short=0 @ 812
  Tag 0xA40A: (0xA40A) Short=0 @ 824
EXIF Interoperability SubSubIFD at offset 7702:
  InteroperabilityIndex: (0x0001) ASCII=R98 @ 7712
  InteroperabilityVersion: (0x0002) Undefined=[48, 49, 48, 48] @ 7724
EXIF MakerNote SubSubIFD at offset 7702:
IFD 1 (Thumbnail) at offset 7732:
  Compression: (0x0103) Short=JPEG Compressed @ 7742
  Orientation: (0x0112) Short=1 @ 7754
  XResolution: (0x011A) Ratio=72 @ 7834
  YResolution: (0x011B) Ratio=72 @ 7842
  ResolutionUnit: (0x0128) Short=Pixels/Inch @ 7790
  JPEGInterchangeFormat: (0x0201) Long=8084 @ 7802
```

```
JPEGInterchangeFormatLength: (0x0202) Long=8830 @ 7814
YCbCrPositioning: (0x0213) Short=2 @ 7826
```

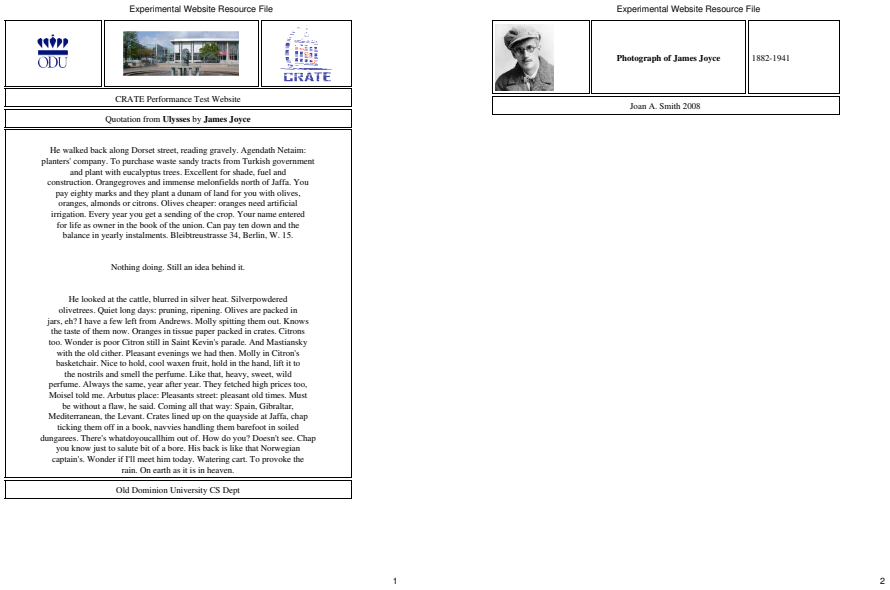


FIG. 84: Sample PDF analyzed by Jhove.

2.2 Jhove With the PDF-Hul

At the time of this writing, there are few non-commercial utilities available for inline analysis of PDF files. One of these is the PDF-Hul for use with Jhove. Figure 84 shows the simple, one-page PDF for which the Jhove analysis is displayed below.

Jhove Output (using PDF-hul)

```
Jhove (Rel. 1.1, 2006-06-05)
Date: 2008-05-14 10:43:18 EDT
RepresentationInformation: /testWeb/group8/pdf93.pdf
ReportingModule: PDF-hul, Rel. 1.5 (2006-03-31)
LastModified: 2008-01-24 13:17:23 EST
Size: 233046
Format: PDF
Version: 1.3
Status: Well-Formed, but not valid
SignatureMatches:
PDF-hul
ErrorMessage: Improperly formed date
Offset: 200
MIMEtype: application/pdf
PDFMetadata:
Objects: 37
FreeObjects: 1
IncrementalUpdates: 1
DocumentCatalog:
```

```
PageLayout: SinglePage
PageMode: UseNone
Outlines:
Item:
Title: Experimental Website Resource File
Destination: 1
Info:
Title: Experimental Website Resource File
Producer: htmldoc 1.8.27
Copyright 1997-2006 Easy Software Products, All Rights Reserved.
ID: 0x9082b9f3aa38c58364dd8e39fe155de9,
0x9082b9f3aa38c58364dd8e39fe155de9
Filters:
FilterPipeline: FlateDecode
Images:
Image:
NisoImageMetadata:
MIMEType: application/pdf
CompressionScheme: Deflate
ColorSpace: palette color
ImageWidth: 60
ImageLength: 72
BitsPerSample: 8
Image:
NisoImageMetadata:
MIMEType: application/pdf
CompressionScheme: Deflate
ColorSpace: palette color
ImageWidth: 496
ImageLength: 192
BitsPerSample: 8
Image:
NisoImageMetadata:
MIMEType: application/pdf
CompressionScheme: Deflate
ColorSpace: palette color
ImageWidth: 88
ImageLength: 99
BitsPerSample: 2
Fonts:
Type1:
Font:
BaseFont: Times-Bold
FirstChar: 0
LastChar: 255
FontDescriptor:
FontName: Times-Bold
Flags: Serif, Nonsymbolic
FontBBox: -168, -341, 1000, 960
FontFile: true
EncodingDictionary:
Differences: true
Font:
```

```
BaseFont: Courier
FirstChar: 0
LastChar: 255
FontDescriptor:
FontName: Courier
Flags: FixedPitch, Nonsymbolic
FontBBox: -12, -237, 650, 811
FontFile: true
EncodingDictionary:
Differences: true
Font:
BaseFont: Times-Roman
FirstChar: 0
LastChar: 255
FontDescriptor:
FontName: Times-Roman
Flags: Serif, Nonsymbolic
FontBBox: -168, -281, 1000, 924
FontFile: true
EncodingDictionary:
Differences: true
Font:
BaseFont: Helvetica
FirstChar: 0
LastChar: 255
FontDescriptor:
FontName: Helvetica
Flags: Nonsymbolic
FontBBox: -174, -285, 1001, 953
FontFile: true
EncodingDictionary:
Differences: true
Pages:
Page:
Label: 1
Page:
Label: 2
```

2.3 Metadata Extractor Utility

Another open source utility which will inspect PDF files is Metadata Extractor Utility from The National Library of New Zealand, used here to inspect the PDF shown in Figure 84.

Metadata Extractor Output

```
<Object><Name>metex</Name><ID>3333</ID>
<ReferenceNumber></ReferenceNumber>
<GroupIdentifier></GroupIdentifier>
<PersistentIdentifier></PersistentIdentifier>
<MasterCreationDate locale="EDT">
<Date format="yyyyMMdd">20080514</Date>
<Time format="HHmmssSSS">104822321</Time>
</MasterCreationDate>
<ObjectComposition>simple</ObjectComposition>
<StructuralType>
<Name></Name> <Extension></Extension>
</StructuralType>
<HardwareEnvironment>i386</HardwareEnvironment>
<SoftwareEnvironment>OS: Linux 2.6.9-67.0.1.ELsmp,
JVM:Sun Microsystems Inc. 1.5.0_07</SoftwareEnvironment>
<InstallationRequirements></InstallationRequirements>
<AccessInhibitors></AccessInhibitors>
<AccessFacilitators></AccessFacilitators>
<Quirks></Quirks>
<MetadataRecordCreator></MetadataRecordCreator>
<MetadataCreationDate locale="EDT">
<Date format="yyyyMMdd">20080514</Date>
<Time format="HHmmssSSS">104822350</Time>
</MetadataCreationDate>
<Comments></Comments>
<Files>
<File xmlns:nz_govt_natlib_xsl_XSLTFunctions=
"nz.govt.natlib.xsl.XSLTFunctions"><FileIdentifier/>
<Path>/var/www/testWeb/group8/pdf93.pdf</Path>
<Filename>
<Name>pdf93.pdf</Name> <Extension>pdf</Extension>
</Filename>
<Size>233046</Size>
<FileDateTime>
<Date format="yyyyMMdd">20080124</Date>
<Time format="HHmmssSSS">131723000</Time>
</FileDateTime>
<Mimetype>application/pdf</Mimetype>
<FileFormat>
<Format>Abobe PDF</Format> <Version>1.3</Version>
</FileFormat>
<Text>
<CharacterSet>ISO-8859-1</CharacterSet>
<MarkupLanguage>unknown</MarkupLanguage>
</Text></File></Files>
</Object>
```

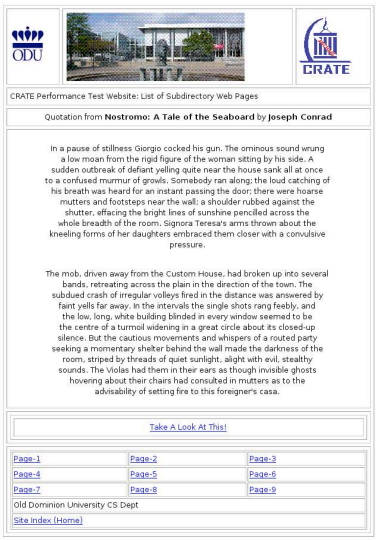


FIG. 85: Sample HTML file for Dublin Core analysis. A view of the HTML file passed to the Dublin Core Utility *jasDC.pl* for analysis

2.4 Dublin Core Utilities

Dublin Core is a common metadata scheme used in Digital Libraries. However, deriving such metadata is not as trivial as it might appear to be. A small utility in Perl was written as part of the CRATE project because no command-line open source tool was available. For this example, one of the test HTML pages is used. An image of the page is shown in Figure 85.

jasDC.pl Output

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
<TITLE>Crate Utility Performance Test
SUBDIRECTORY INDEX for /home/jsmit/testWeb/group12/dir2
</TITLE>
<META http-equiv="Content-Type" content="text/html"; charset="UTF-8">
<META content="links to site subdirectories" name="DC.source" >
<META content=" Crate Utility Performance Test
SUBDIRECTORY INDEX for /home/jsmit/testWeb/group12/dir2 INDEX"
name="DC.title" >
<META content=" Joan A. Smith" name="DC.creator" >
<META content="2008-1-24" name="DC.date" >
<META content="Links to Subdirectory indices"
name="DC.identifier" >
<META content="Copyright ODU" name="DC.rights" >
<META content="Page produced as part of performance test"
name="DC.description" >
```

2.5 Open Text Summarizer

The Open Text Summarizer utility attempts to distill a brief summary of a text-based document. It appears to "weight" words by frequency and as a result will misinterpret tags in, for example, HTML documents as being "important" words. Using the "-a" (about) option on even a lengthy file, it returns a short, descriptive phrase, for example:

Article talks about "web,resources,metadata,files,typical"

is the full literal text output from processing the article cited in [173].
In contrast, using the default option which summarizes the text, a lengthier overview of the content is produced. The following summary is taken from a text version of a D-Lib Magazine article [173], stripped of its HTML tags. The utility selects what it considers to be the "important" phrases or sentences from the document and outputs them as a single, unbroken line of text. Since punctuation may be left off, particularly if the phrase is a heading rather than a sentence, the result is not necessarily grammatically nor syntactically correct and may read awkwardly. However, the purpose is *distillation* for preservation, not for publication per se.

The Derived Summary

```
We propose a simple model for such everyday websites which takes
advantage of the web server itself to help prepare the site's
resources for preservation.  The web server responds to the
archiving repository crawler by sending both the resource and the
just-in-time generated metadata as a straight-forward
XML-formatted response. Even though digital libraries are often
accessed as websites, anyone involved with digital libraries can
easily point out the many differences between everyday websites
and a true Digital Library (DL). The Web is an unorganized
amalgamation of digital pages with little metadata and
unpredictable additions, deletions, and modifications âĂŞ a
crawlapalooza for the web robot. For an archiving repository
seeking to preserve websites, the site preparation process is
challenging thanks to the wide variety of resource types and
content that exist on websites. Typically, an archivist will
crawl the target website then process each resource with various
metadata utilities to extract technical information. From the
webmaster's point of view, the ideal solution would be a tool
installed on the web server which manages itself, and which
automatically provides the "extra information" (i.e., metadata)
that the archiving site needs to prepare the website for
preservation, and which does not impact the normal operation of
the web server. Motivation We begin by observing that digital
preservation: remains in the niche of librarians and
archivists is not sustainable as an ex post, ad hoc process it
should be congruent with practices of the general web community
it is applicable to content whose value is not always known in
advance As participants of the Archive Ingest and Handling Test
```

(AIHT), 1 one of the lessons we learned was that preserving the GMU 911 website was, in a sense, made more difficult because the website was not harvested directly from the web, but rather processed by site administrators and given directly to us. The purpose of the research reported in this article is to build a framework that allows dual access to web resources: the existing HTTP access mechanisms for conventional web agents, and a "preservation-ready" access channel that integrates the best tools of the digital preservation community into the web server. A typical HTTP response contains just enough information to enable the smooth transfer of content from web server to web client or crawler. In short, web server MIME typing has serious limitations when it comes to providing adequate preservation information about the data format of web resources. Clearly, getting the responding server to preprocess the resource and include the results together with the original resource in one complex-object response would help both the particular archivist and the general goal of web preservation. Metadata Utilities How can metadata be derived for web resources? Name Description Jhove Analysis & characterization by type (img, audio, text) Kea Key phrase extraction OTS Open Text Summarizer ExifTool Image/video metadata extractor PDFlib-pCOS Extract PDF metadata (commercial tool) MP3-TAG Extract audio file tags Essence Customized information extraction GDFR Extended MIME file typing MD5 Message Digest File Magic Type identification using special ID bits of the file DROID File signature analysis (internal and external) Table 2: Some utilities for producing resource metadata. If we combine this output with a Base64-encoding of the resource, we would have a neatly packaged, pre-processed web resource ready for archive ingestion and preservation preparation. The concept calls for the disseminating web server to preprocess the resources it serves up by using metadata-generation utilities, such as those described here, and to serialize this information together with the Base64-encoded resource in a simple XML-formatted complex object response. They are specified in the web server's configuration file, using a simple enabling directive: LoadModule <module-name> <path/to/module.so> Common examples for the Apache web server are mod_perl and mod_python (Perl/Python-CGI optimizers), mod_ssl (to support secure socket layer connections), and mod_jserv (Java servlet engine). Figure 2: (A) Normal web page request and (B) OAI-PMH request For the example given in Figure 2-B, the \textsc{modoai} response is returned as human-readable ASCII in the DIDL XML format (called a "DID"), with the resource encoded in Base64 and with the HTTP response headers included as basic metadata (see Table 3, below). The utilities enabled on this server are: File - Looks at the "magic bytes" to determine MIME type MD5 - Provides the MD5 hash value of the file Jhove - File analyzed using the HUL appropriate to specific file type DROID - Pronom's DROID utility which evaluates MIME type ExifTool - Phil Harvey's Perl script which analyzes images Best-effort Metadata The approach described here is a best-effort approach to metadata.

APPENDIX E

APACHE CONFIGURATION DIRECTIVES FOR MODOAI

1 THE STRUCTURE OF THE MODOAI CONFIGURATION FILE

Apache uses a series of configuration commands usually located either in a single file ("httpd.conf") or in a series of files ("apache2.conf", "ports.conf", "modoai.conf", etc.). Each line in a configuration file is equivalent to a "command" or "directive" which controls Apache's responses to HTTP events. Most modules – mod_rewrite, mod_perl, and mod_cgi for example – have numerous switches or configuration options that can be individually declared or suppressed. This allows the webmaster to completely customize the installation and behavior of Apache for any site.

An example configuration for MODOAI is shown in Appendix E–2. The individual lines have been numbered for ease of reference. Each number in the left margin indicates the start of a new, single logical line (in the actual Apache conf file, there are no line numbers). In addition, breaks within a line (i.e., an EOL character) are not allowed, so in the configuration file the string must occur on a single text line.

A *whitespace* character acts as the delimiter within a single line. Apache reads this section into a structure accessible by Apache and the module. Lines 1 – 3 define the scope and handling of the directives for the MODOAI module; Table 32 explains each of these lines.

The remaining configuraton lines create and populate variables used by the MODOAI module. For example, lines 4 –7 define the variables "modoai_sitemap" (location and name of the Sitemap file), "modoai_admin" (user account associated with MODOAI administration), and "modoai_email" which can be called by name from within MODOAI. These variables are simple and unique, i.e., only one of each exists in the module.

For variables needing multiple definitions, the Apache API provides a 5-part structure consisting of the variable name and up to 4 attributes. This structure allows the module to have any number of items with the same variable name. Using this signature, more than one metadata utility can

TABLE 32: Directives in modoai.conf

#	Directive	Explanation
1	Alias /modoai "/var/www/"	MODOAI points to webroot
2	<Location /modoai>	if home URL ends with /modoai, then…
3	SetHandler modoai-handler	use the MODOAI module

be named. Thus, lines 11 – 24 create a single variable called *modoai_plugin* with 11 instances of the variable defined. Each entry has 4 elements, delimited by a *whitespace* character. Quotations surround any element that requires whitespace within it, such as command-line arguments for a utility. Where the utility itself also expects in-line quotations, these are embedded within outer quotes of the element.

Each *modoai_plugin* first element is a *tag name*. It can be whatever the webmaster wants to call it – MODOAI does not refer to the tag name per se. Since the tag name will appear in CRATE responses, the recommended tag name for each modoai_plugin variable is the short name of the metadata utility executable.

The modoai_plugin variable's other three elements assigned are, in order of appearance: (2) the execution command (3) the command to get utility version information and (4) the Mime types for which the utility is invoked. As an example, lines 11 (the Word Count utility) and 21 (the Exif Tool utility) can be interpeted as shown in Table 33. Appendix E–2 contains the actual configuration directives used for the CRATE utilities experiments discussed in Chapter IX.

TABLE 33: MODOAI plugin elements & attributes

modoai_plugin	*wc*	*/usr/bin/wc %s*	*/usr/bin/wc –v*	*text/**
add a plugin to the list	name is "wc"	executable command %s: filename	command to print version info	use on any text file
modoai_plugin	*exifTool*	*/usr/bin/exiftool -a -u %s*	*/usr/bin/exiftool –v*	*image/**
add a plugin to the list	name is "exifTool"	command; -a,-u are switches; %s: filename	command to print version info	use on any image file

Some utilities have a wide variety of switches which can be applied to particular combinations of Mime types. In this case, it may be simpler to create a script to address each particular situation. Jhove is an example of a utility which has so many options that a script could be a useful implementation method. The actual Jhove script used for the CRATE experiments is given in Appendix E–3. Alternatively, each Jhove call in the script could exist as a unique plugin variable in the configuration file. Each method has its pros and cons in terms of maintenance simplicity, ease of implementation and the administrative approach of the local webmaster. Overhead cost, of course, will vary from system to system.

Finally, responses generated by MODOAI can be very large. The variables on lines 9 and 10 are used by the server to help control response size - in bytes (max_response_size) and in absolute number of records per response (max_response_items). They determine when a *Resumption Token* needs to be issued to the harvester. Setting the number very large as in the example configuration file is equivalent to having no upper limit on the variable(s). In implementation, whichever limit is reached first is the one that applies. If the size limit in bytes is reached before the item count

limit, that portion of the response completes. That is, no response contains an incomplete item. Because the response is built *in situ*, the final size of any record in the response is not known until it has been processed by each of the subcomponents - plugin, XML writer, subrequest handler, etc. - of MODOAI. Similarly, a single GetRecord request could produce a response that exceeds the max_response_size parameter, but it would still be provided to the harvester in one response event, that is, no Resumption Token would be needed nor issued.

2 CONTENTS OF THE MODOAI.CONF FILE

```
 1  Alias /modoai "/var/www/"
 2  <Location /modoai>
 3    SetHandler  modoai-handler
 4    modoai_sitemap  /var/www/sitemap.xml
 5    modoai_admin  jsmit
 6    modoai_email  admin@crate.gotdns.com
 7    modoai_gateway_email  mail@crate.gotdns.com
 8    modoai_oai_active ON
 9    modoai_max_response_size  9999999999
10    modoai_max_response_items 9999999999
11    modoai_plugin wc '/usr/bin/wc %s' '/usr/bin/wc --version' text/*
12    modoai_plugin file '/usr/bin/file %s' '/usr/bin/file --version' */*
13    modoai_plugin md5sum '/usr/bin/md5sum %s'
      '/usr/bin/md5sum --version' application/*
14    modoai_plugin sha1sum '/usr/bin/sha1sum %s'
      '/usr/bin/sha1sum --version' image/*
15    modoai_plugin sha224sum '/usr/bin/sha224sum %s'
      '/usr/bin/sha224sum --version' image/png
16    modoai_plugin sha384sum '/usr/bin/sha384sum %s'
      '/usr/bin/sha384sum --version' image/jpeg
17    modoai_plugin sha256sum '/usr/bin/sha256sum %s'
      '/usr/bin/sha256sum --version' application/pdf
18    modoai_plugin zipinfo '/usr/bin/zipinfo %s'
      '/usr/bin/zipinfo --version' */zip
19    modoai_plugin jhove "/opt/jhove/jhove -c
      /opt/jhove/conf/jhove.conf
      -m jpeg-hul -h xml %s" "/opt/jhove/jhove
      -c /opt/jhove/conf/jhove.conf -h xml -v" image/jpeg
20    modoai_plugin pronom_droid "/opt/jdk1.5.0_07/bin/java -jar
      /opt/droid/DROID.jar
      -L%s -S/opt/droid/DROID_SignatureFile_V12.xml"
      "/opt/jdk1.5.0_07/bin/java -jar
      /opt/droid/DROID.jar -V" */*
21    modoai_plugin exifTool "/usr/bin/exiftool -a -u %s"
      "/usr/bin/exiftool -ver" image/*
22    modoai_plugin ots '/usr/local/bin/ots -a %s'
      '/usr/local/bin/ots -v' text/*
23    modoai_plugin metadata-extractor
      "/home/jsmit/metadata-extractor/dist/extract.sh extract
        'NLNZ Data Dictionary' Default simple 'Blah' 3333 %s
        2>/dev/null %s"
      "/bin/echo metadata-extractor 3.4" image/jpeg
24    modoai_plugin dcdot '/opt/dublinCoreUtil/dcdot/jasDC.pl %s'
      '/opt/dublinCoreUtil/dcdot/jasDC.pl --version' text/html
25  </Location>
```

3 EXAMPLE SHELL SCRIPT INVOKING JHOVE OPTIONS

```
#!/usr/bin/perl -w
$|=1; #piping-hot I/O

use strict;
use File::Copy;

if ($#ARGV !=0){
       die "\nUsage: procJhove.pl path/to/file/filename \n\n";
}

my $fileName = $ARGV[0];
my $format = `file -b $fileName`; #-i (mime) produces inadequate typing
my $module = "BYTESTREAM"; # Module: BYTESTREAM 1.2 (default)

#Jhove module processing types installed

#text processors:
  $module = "WAVE-hul" if ($format=~m/wmv/i);# Module:WAVE-hul 1.2
  $module = "ASCII-hul" if ($format=~m/ascii/i);# Module: ASCII-hul 1.2
  $module = "HTML-hul" if ($format=~m/html/i);# Module: HTML-hul 1.2
  $module = "UTF8-hul" if ($format=~m/utf\-8/i);# Module: UTF8-hul 1.2
  $module = "XML-hul" if ($format=~m/xml/i);# Module: XML-hul 1.3
  if (($format=~m/text/i)&&($format=~m/script/i)){
   $module = "ASCII-hul"; #plain scripts usually ascii
   }

#image processors:
  $module = "AIFF-hul" if ($format=~m/aiff/i); # Module: AIFF-hul 1.3
  $module = "GIF-hul" if ($format=~m/gif/i);# Module: GIF-hul 1.2
  $module = "JPEG-hul" if ($format=~m/jpeg/i);# Module: JPEG-hul 1.2
  $module = "JPEG2000-hul" if (($format=~m/jpeg/i)&&($format=~m/2000/));
  $module = "TIFF-hul" if ($format=~m/tiff/i);# Module: TIFF-hul 1.4

#application processors:
  $module = "PDF-hul" if ($format=~m/pdf/i);# Module: PDF-hul 1.5

#audio processors:
  $module = "WAVE-hul" if ($format=~m/wav/i);# Module: WAVE-hul 1.2

#print "\nFile: \t$fileName\t";
#print "info: \t$format";
#print "Module: $module\n";

my $cmd = "/opt/jhove/jhove -c /opt/jhove/conf/jhove.conf -m ";
$cmd = $cmd." $module $fileName";
my $jhove = `$cmd` ;
print "$cmd\n";
exit;
```

APPENDIX F

SITEMAP FILES

1 USING SITEMAP TOOLS

There are a number of tools for webmasters to use which will help to build a sitemap file such as [9, 67] which are installed and run locally. Since they are "privileged" applications, they can mine databases and web server logs for more complete site coverage. Others run on remote servers. [8, 199] These accept a starting URL and proceed to follow all of the links found (Figure 86) in pages on the site. Most tools will also test each of the links, noting those which are found (HTTP Response Code = 200) versus those that are broken (HTTP response Code = 404). Obvious duplicates are removed, and the list of links is converted into an XML document in accordance with the sitemaps protocol. [167] Thus, these links:

```
http://localhost/testWeb/group8/pdf120.pdf
http://localhost/testWeb/group8/pdf1.pdf
http://localhost/testWeb/group8/dir3/pg5.html
```

will become the following entries in a sitemap file:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
   <url>
      <loc>http://localhost/testWeb/group8/pdf120.pdf</loc>
   </url>
   <url>
      <loc>http://localhost/testWeb/group8/pdf1.pdf</loc>
   </url>
   <url>
      <loc>http://localhost/testWeb/group8/dir3/pg5.html</loc>
   </url>
</urlset>
```

The sitemaps protocol *requires* the XML and namespace declarations, as well as a <loc> (location) tag (within <url> and </url> tags) for each URL. There are a few other tags that are *recommended*, however. Part of the information available to the crawler includes the timestamp on the file. This information is added to the sitemap within "last modified" (<lastmod>) tags:

```
<url>
   <loc>http://localhost/testWeb/group8/pdf120.pdf</loc>
```

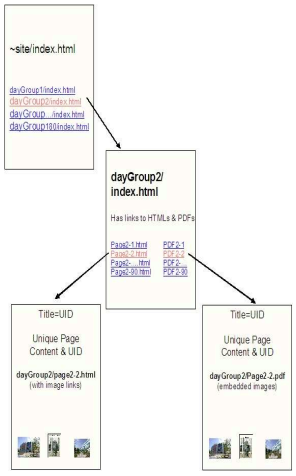


FIG. 86: Links on a web page provide crawlers with a list of resources to ask for. Usually only *internal* links are followed.

```
   <lastmod>2007-11-21T14:35:21Z</lastmod>
</url>
```

Two other useful tags are the *priority* and *change frequency* tags. Priority can range from 0.0 to 1.0, and is used to by search engines to determine the order of pages in a query result set (it does not impact how the site ranks compared to other sites). The protocol specifies a default URL a priority of 0.5, which can be manually edited by the webmaster. Change frequency signals how often crawlers can expect to find new information at that particular link. Any combination of these tags (provided the minimum set is present) can be specified on a per-link basis. Sitemap tools will typically ask what frequency should be assigned to pages on the site, since no default is specified by the protocol. There are specific values allowed, ranging from “always” to “never.” A common value is “monthly.”

The last item that is required is UTF-8 encoding of the ampersand, quotes (single and double), and the “<” and “>” symbols. Thus the URL:

`http://localhost/testWeb/knitting&cat_01.png`

will become:

`<loc>http://localhost/testWeb/knitting&amp;cat_01.png</loc>`

in the sitemap file.

2 EXAMPLE SITEMAP FILE

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9"
     url="http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd"
     xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
<url>
   <loc>http://localhost/index.html</loc>
   <lastmod>2007-10-20T11:03:00Z</lastmod>
   <priority>1.000</priority>
   <changefreq>monthly</changefreq>
</url>
<url>
   <loc>http://localhost/testWeb/group8/pdf120.pdf</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
</url>
<url>
   <loc>http://localhost/testWeb/group8/pdf1.pdf</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/pg5.html</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
   <priority>0.5000</priority>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/pg4.html</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
   <priority>0.5000</priority>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/index.html</loc>
   <lastmod>2008-01-01T04:05:01Z</lastmod>
   <priority>1.0000</priority>
   <changefreq>monthly</changefreq>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/knitting_01.png</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
   <priority>0.5000</priority>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/drink_icon_01.gif</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/pg9.html</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
   <priority>0.5000</priority>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/pdf94.pdf</loc>
```

```
   <lastmod>2007-11-21T14:35:21Z</lastmod>
   <priority>0.7000</priority>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/alice05a.png</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
   <priority>0.5000</priority>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/pg8.html</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
   <priority>0.5000</priority>
</url>
<url>
   <loc>http://localhost/testWeb/group8/dir3/pg1.html</loc>
   <lastmod>2007-11-21T14:35:21Z</lastmod>
   <priority>0.8000</priority>
</url>
</urlset>
```

VITA

Joan A. Smith
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

EDUCATION

Ph.D. Computer Science, Old Dominion University, 2008
M.A. Computer Education, Hampton University, 1988
B.A. Natural Science, University of the State of New York, 1986

PROFESSIONAL EXPERIENCE

2008–Present Chief Technology Strategist, Emory University Libraries
2004–2008 Research Assistant, Old Dominion University
2000–2004 Business Owner and Consultant
1998–2000 Northrop Grumman, Inc.
1989–1998 Inter-National Research Institute
1987–1989 Electronic Institute of Technology

PUBLICATIONS AND PRESENTATIONS

A complete list is available at `http://www.joanasmith.com/pubs.html`

PROFESSIONAL & ACADEMIC SOCIETIES

Association for Computing Machinery (ACM)
Institute of Electrical and Electronics Engineers (IEEE)
Alpha Gamma Sigma National Honor Society ($A\Gamma\Sigma$)
Golden Key International Honor Society

PERMALINKS

Home Page: `http://www.joanasmith.com/`
Email: `mail@joanasmith.com`

Typeset using LaTeX.