

Summer 2008

Integrating Preservation Functions Into the Web Server

Joan A. Smith
Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/computerscience_etds



Part of the [Computer Sciences Commons](#), and the [Digital Communications and Networking Commons](#)

Recommended Citation

Smith, Joan A.. "Integrating Preservation Functions Into the Web Server" (2008). Doctor of Philosophy (PhD), dissertation, Computer Science, Old Dominion University, DOI: 10.25777/jr7m-6g71
https://digitalcommons.odu.edu/computerscience_etds/22

This Dissertation is brought to you for free and open access by the Computer Science at ODU Digital Commons. It has been accepted for inclusion in Computer Science Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

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 descriptive metadata available for the resources is so minimal that it is sometimes insufficient for a browser to recognize. 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 both knows 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. The CRATE Model defines a method for addressing the Counting Problem and the Representation Problem using existing web server-compatible technology. A series of experiments which evaluated this approach 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.

©Copyright, 2008, by Joan A. Smith, All Rights Reserved.

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 plain 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. 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 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 and Aravind Elango, who wrote the initial prototype of MODOAI, from which the author's current modular version was derived.

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 such an academic endeavor 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 moral support to technical opinions his help has been invaluable.

Michael Nelson guided the development of the CRATE concept and its focus on the *counting* and *description* 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. This PhD belongs to him.

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)
DL	Digital Library
GIF	Graphics Interchange Format
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JPEG	Joint Photographic Experts Group
KWF	Kahn-Wilensky Framework
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	Uniform 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		xi
LIST OF FIGURES		xiii
CHAPTER		
I	INTRODUCTION	1
	1 The Challenge of Digital Preservation	1
	2 Scope	4
	3 Approach	5
	4 Organization	7
II	CURRENT PRACTICE IN DIGITAL PRESERVATION	9
	1 Preservation: Definitions and Limitations	9
	2 Digital Preservation Models & Implementations	12
	3 LOCKSS	19
	4 OAI-PMH	21
	5 Socio-Economic Factors Influencing Preservation	23
	6 Summary	25
III	THE CURRENT ROLE OF THE WEB SERVER IN DIGITAL PRESERVATION	26
	1 Incidental Web Preservation: Usenet	26
	2 Intentional Web Preservation: Internet Archive	27
	3 Other Web Archiving Efforts	31
	4 The Web Server As Agent of Migration	32
	5 Summary	34
IV	THE CURRENT ROLE OF SEARCH ENGINES IN DIGITAL PRESERVATION	35
	1 The Search Engine as Agent of Discovery	35
	2 The Search Engine as Agent of Refreshing	52
	3 The Search Engine as Agent of Preservation	55
	4 Summary	55
V	RESOURCE ENUMERATION: THE COUNTING PROBLEM	56
	1 The Counting Problem Defined	56
	2 Why the Counting Problem Exists	56
	3 The Sitemap Protocol	61
	4 Summary	63
VI	EVALUATION OF RESOURCE ENUMERATION METHODS	64
	1 A Counting Problem Experiment	64
	2 A Comparison of Enumeration Methods	76

3	Summary of Experiment Results	82
4	Resource Enumeration & the Race Condition Problem	82
5	Strategies for Optimizing Resource Enumeration	83
6	Summary	83
VII	RESOURCE DESCRIPTION: THE REPRESENTATION PROBLEM	84
1	The Representation Problem Defined	84
2	Why The Representation Problem Exists	85
3	Search Engines & Representation	86
4	Web Servers, Browsers, & Representation	88
5	Representation Models, Metadata, and Interoperability	91
6	Summary	92
VIII	EVALUATION OF METADATA UTILITIES ON THE WEB SERVER	93
1	A Representation Problem Experiment	93
2	Performing the Experiments	105
3	A Quantitative Comparison of Utility Performance	106
4	Summary of Experiment Results	109
5	Strategies for Selecting Metadata Utilities	110
6	Summary	110
IX	CRATE: A MODEL FOR SELF-DESCRIBING WEB RESOURCES	112
1	CRATE: A Data-Centric Preservation Model	113
2	Complex Objects As Archival Information Packages (AIPs)	114
3	Building The CRATE	118
4	CRATE Compared with Other Complex-Object Models	119
5	Implementing the CRATE Model	120
6	Summary	122
X	MODOAI: AN IMPLEMENTATION OF THE CRATE REFERENCE MODEL	123
1	Background: The Apache Web Server	123
2	History of MODOAI	127
3	The Design & Structure of MODOAI	129
4	Customizing MODOAI	133
5	CRATE Deployed In MODOAI	134
6	Summary	134
XI	CONCLUSIONS	135
1	Contributions	135
2	Future Work	137
3	Integrating Preservation Functions into the Web Server	138
	BIBLIOGRAPHY	140
	APPENDICES	
A	CRATE XML SCHEMA DOCUMENTS	153
1	The Simple CRATE Schema: crate.xsd	153

2	CRATE LANL DIDL Schema: oaicrate.xsd	155
B	EXAMPLE CRATE RESPONSES	156
1	CRATE Plugins in the Identify Response	156
2	CRATE Plugins in the Get Record Response	160
3	OAI-PMH List Identifiers Response	163
C	EXAMPLES OF METADATA UTILITY OUTPUT	164
1	Comparative Metadata Output of a Small JPEG File	164
2	Other Examples of Metadata Utility Output	169
D	APACHE CONFIGURATION DIRECTIVES FOR MODOAI	178
1	The Structure of the MODOAI Configuration File	178
2	Contents of the modoi.conf File	181
3	Example Shell Script Invoking Jhove Options	182
E	SITEMAP FILES	183
1	Using Sitemap Tools	183
2	Example Sitemap File	185
VITA	187

LIST OF TABLES

Table	Page
1 Threats to Digital Preservation	3
2 Dublin Core Categories	11
3 File & Structural Map Sections of METS	14
4 OAI-PMH Verbs	22
5 Problematic snapshots on the Wayback Machine	29
6 Website content of the Sliding Directories experiment	38
7 Log data from two sites	41
8 Crawler Statistics from the Sliding Directories experiment.	42
9 HTTP Request-Response Example	60
10 Composition of the CS Website	65
11 MIME Distribution on Test Website	66
12 Web Server Log Fields	70
13 HTTP Response Codes	73
14 Request Distribution by HTTP Method	73
15 Request Distribution by MIME Type	74
16 Request distribution on the CS Website	74
17 Resource duplication on a site	75
18 Links on CS Website Main Page	78
19 Fully-qualified internal links on the CS Website.	79
20 Dynamic Resource Request Distribution	82
21 The MIME Content Type Categories	90
22 HTTP Headers	91
23 Content Distribution on Test Website	97
24 Test Website MIME Type Resource Distribution	97
25 Metadata Utility Assessment	100
26 Plugin Specification	101
27 Average distribution of hits per test run.	107
28 Performance Metrics	108
29 Directives in modoai.conf	178
30 MODOAI plugin elements & attributes	179

LIST OF FIGURES

Figure	Page
1	Preservation vs Backup 2
2	Website as tree 5
3	Conceptualization of the Kahn-Wilensky Framework 10
4	OAIS Functional Model 13
5	VERS, METS & PREMIS Complex Objects 15
6	The VERS Workflow 17
7	The LANL MPEG-21 DID Complex Object 18
8	Acroread Metadata 20
9	Basic OAI-PMH Data Model 21
10	OAI-PMH With Complex Objects 24
11	Languages on the Rosetta Stone 27
12	Archived CS-ODU Webs on Wayback 28
13	Jhove Metadata 30
14	ARC Model 31
15	Unhandled MIME Type & Resource Migration 33
16	Number of resources in the test website 37
17	Sliding Directories structure 38
18	Gradual resource removal 40
19	Crawling patterns on site MLN 43
20	Hourly crawling patterns on site MLN 44
21	Deep website schema 46
22	Entry page to test website 46
23	Wide & Deep Sites 47
24	Subdirectory Main Page Examples 49
25	Crawling Patterns on Bread Crumb Site 50
26	Crawling patterns on Buffet Site 51
27	Buffet Site Crawling Pattern 53
28	Bread Crumb Site Crawling Pattern 54
29	Website & Apache Server File System 57
30	Crawler's view of a website 60
31	Website URLs in a Sitemap file 62
32	The CS Department Website 67
33	Web Log Gaps 2006 68
34	Web Log Gaps 2007 69
35	Percent of All Resources Visited 70
36	Percent of Public Resources Visited 71
37	Resources Found in Logs & Snapshot 77
38	Snapshot Resources Crawled 79
39	Report from Audit My PC 80
40	Site Coverage from Web Logs 81
41	Website Coverage from Integrated Counting Techniques 82
42	Resource Representation 84
43	Representing Content 87

44	ASCII Art	88
45	Search Engine resource transformation	89
46	Test Website Sample Page	95
47	PDF from the Test Website	95
48	Visual Map of the Test Website	96
49	META Tag Content Example	99
50	Apache Configuration	102
51	Web Traffic Patterns	103
52	Sample CRATE XML	106
53	CRATE Process in OAIS Context	113
54	CRATE in the OAIS Model	115
55	Two Views of a Resource	117
56	The CRATE model and example CRATE configurations	118
57	CRATE Object as MPEG-21 DID	121
58	The Apache Server Life Cycle	124
59	The Apache Process Loop	126
60	The MODOAI processing loop	131
61	Two foos having lunch	165
62	KDE Desktop File Inspector	168
63	A Sample Digital Photograph for Exif Tool	169
64	Sample PDF analyzed by Jhove	171
65	Sample HTML file for Dublin Core analysis	175
66	Links on a web page	184

CHAPTER I

INTRODUCTION

Digital information lasts forever or five years, whichever comes first.

— Jeff Rothenberg[118]

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 [95], the UK [93], Holland [66], Australia [94], and Japan [142], 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 [59], but so far no international standard has been developed and accepted [73].

Preserving “everyday” websites is arguably harder than preservation of more formal collections 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, and a single site 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 by, for example, topic.

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 sites have in common is that they are hosted on a web server. Since web servers typically host many sites, 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, backup facilities - all of which are automated features requiring little to no manpower support by the provider. What if there were an automated preservation option (as opposed to merely a site back-up process), which is similarly simple to 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.

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



FIG. 1: 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 that the Rosetta Stone [115] in the British Museum (left) is the “original” while the item for sale in a museum catalog [116] on the right is a *replica*, i.e., not the original.

There is a difference between *preservation* and *backup*, although to the lay person that difference is perhaps not as obvious in digital media as it is in archeology (see Figure 1). 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 particular file is the original and the others are copies. The distinction becomes less clear if the copies are made with the “preserve timestamp” option.

Since digital media is so readily replicated, the need for a preservation strategy may not be obvious. There are many threats to the long-term persistence of digital information. Rosenthal et al. [114] identified over a dozen factors that could limit or prevent recovery (see Table 1 on the next page). Examples of each can be found in world news archives. In the digital world, preservation encompasses not just *storing* bits and bytes but also ensuring that they continue to be *accessible*, whether they are dark archives (for example, backup tapes which may be only rarely accessed, or historical files retained for general archival purposes), or a website’s often-retrieved home page.

Arguably the most influential document on digital preservation has been the *Reference Model for an Open Archival System* [20]. A comprehensive review of issues and best practices written by

TABLE 1: This list of threat factors for digital preservation comes from [114]. 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 | |

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

All three are essential for preserving digital information. 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.

Maintaining a viable archive is only part of the story, however; the content should also be discoverable. On the web such 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 [16], wasting computational cycles and bandwidth. A bigger problem is that large sections of the web remain “hidden” [110], 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 [87].

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 [12, 56, 27], cached by search engines [87], haphazardly stored by casual users, or intentionally archived as part of a site snapshot [146, 61]. 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”) [86]. 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 the interaction between web

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

This is a much harder problem to solve than it would seem at first glance. The answer depends on who is asking the question. *Content-negotiation*, for example, can determine whether a document is returned in an English-language version or a French-language version. Some parts of a site may be cloaked from certain clients. Yet other difficulties stem from the fact that a website is not a simple mapping of a file system tree (Figure 2 on the following page). Dynamic content like CGI scripts also adds to the problem's complexity. In addition, web resources may not physically exist on the server but instead be composed on-the-fly by the combination of a file system resource (HTML page), data from a database (dynamic content) and the access control imposed by the web server configuration file (`httpd.conf`, for instance). A subsequent visitor to the site who issued the identical request string could actually receive different content in reply. For example, if the dynamic content is a request for Unix time (number of seconds since 1970), that portion of the page will not be the same as the previous response. Even though a website is visualized as a directed graph of URIs, a URI does not necessarily have a file system equivalent. The loose hierarchy of web pages, including the hypothetical "website tree" pictured in Figure 2 on the next page, are abstractions of reality. Web resources exist as URIs, essentially independent of any links to other URIs. The mapping between a website and the file system is thus not one-to-one. In short, enumerating every accessible site resource may not be possible but we may be able to utilize features of the file 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** (getting all of a website's resources) and the **Representation Problem** (getting sufficient resource description information, i.e., "metadata"). Both are crucial for digital preservation [117], 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 practical and safe 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?

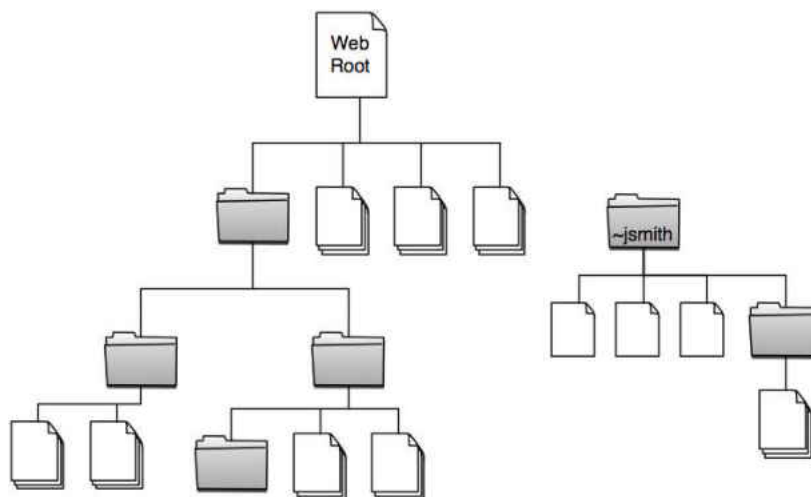


FIG. 2: Websites are often described as “trees”, with website “root” (index.html) linking to resources in other parts of the site. In many cases, however, resources are *not* linked to the main part of the website (lower right). In other cases, they may link back-and-forth: websites are more like graphs than trees. This is one reason why a website “counting problem” exists.

(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 web sites. 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.
 - *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. It 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-size site having a variety of content commonly found on websites (HTML, PDF, Video, etc.). Detailed performance metrics were obtained for the utilities.
 - *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. The reference model for this approach is called CRATE. A technical implementation of the CRATE reference model was developed and evaluated via a new software module, MODDOI. 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 MODDOI module has been installed on several Apache web servers and tested under different load scenarios. MODDOI 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-size 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 meta-data).

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 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 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: 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 9: 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 10: MODOAI: An Implementation of the CRATE Reference Model Describes the design, development, and implementation of an Apache web server module which implements the CRATE model.

Chapter 11: Evaluation of the MODOAI CRATE Implementation Reviews the results of the experiments on the Counting Problem and the Description Problem, and evaluates the feasibility of integrating preservaton functions into the web server.

Chapter 12: Future Work This chapter presents follow-on research areas suggested by the results of the work presented in this dissertation.

Chapter 13: Conclusions 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: Provides 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.

CHAPTER II

CURRENT PRACTICE IN DIGITAL PRESERVATION

Data should be “born archival”

— Stewart Brand[15]

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, and an expected *persistence* over time thanks to *preservation policies* and procedures. Web sites also have a specific path to each resource, but 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 web site 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) [63], originally published in 1995 [62]. 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 3 on the following page illustrates these concepts.

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 (handles), ICANN and DNS (handle authorities), an individual web site is often disorganized, lacks metadata, has no inherent rights-management structure, and may literally disappear overnight. Despite superficial similarities, digital libraries and the world wide web have very different characteristics.

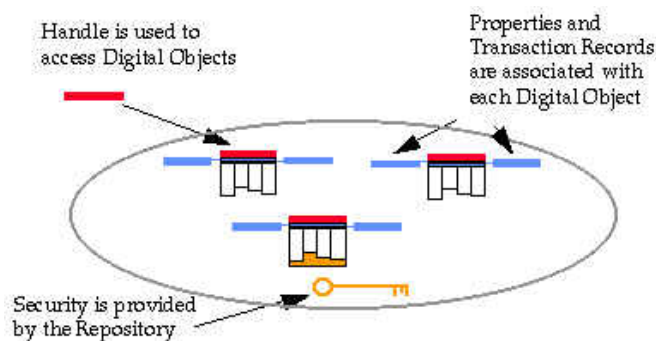


FIG. 3: A digital library (repository) as conceived in the Kahn-Wilensky Framework. Image taken from [3].

1.2 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 any repository to implement [31]. Like many metadata schemes, Dublin Core (“DC”) evolved over time and now includes dozens of fields, elements, qualifiers, and options. Table 2 on the next page 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 [70, 21]. 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” (AACR2, for example) are part of Library Science curricula at universities nationwide. Metadata content is still a field for experts.

1.3 The Resource Description Framework

The Resource Description Framework (RDF) evolved alongside Dublin Core [89, 111], but exists independently. It is “designed to encourage the reuse and extension of metadata semantics” [89] and as such can contribute to the metadata available for a particular resource. RDF is a W3C recommendation, and part of the semantic web standard, an effort which seeks to map a “relationship web” onto the Internet. RDFa [141] 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-scrappable. Like Dublin Core, RDF calls for informed participation rather than amateur input which makes it less likely to be used on pedestrian web sites, and which is more typically implemented as part of a professional digital library such as Connexion [29]. Dublin Core, RDF, and other metadata schemes have been incorporated into HTML/XHTML – for example, by using

TABLE 2: An example using 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 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

META tags (Dublin Core) or RDFa – but usage outside of digital libraries is not yet common [46].

2 DIGITAL PRESERVATION MODELS & 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 [28]. 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 [20]. 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 [20] 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 4 on the following page, 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

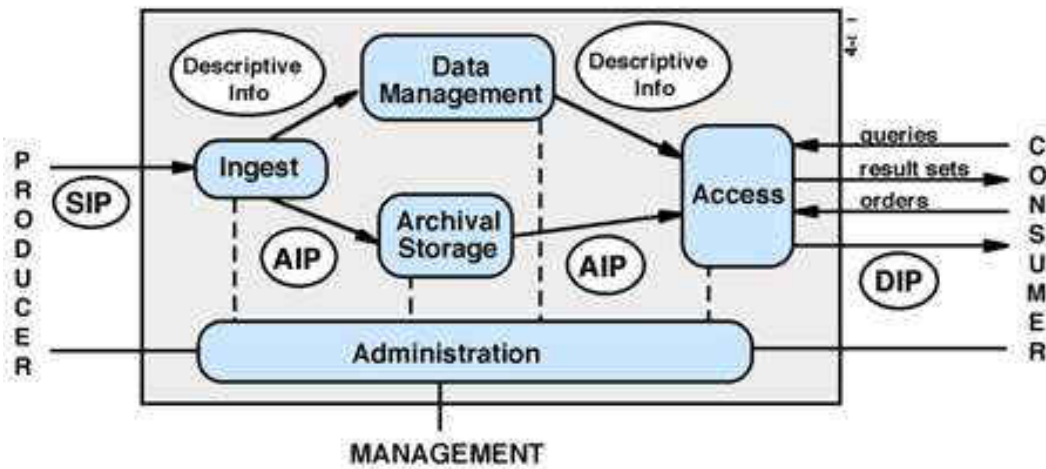


FIG. 4: This view of the OAIS Functional Model is taken from the CCSDS Report ([20]). Note the operationally-intense role of management in creating and maintaining the Archival Information Packages (AIP).

physical and digital objects, the harder problem is acknowledged to be in the digital realm [117] as anyone with a 10-year-old floppy disk and a 100-year-old family bible can readily understand.

Archival Information Packages (AIPs) are expected to contain sufficient (ideally, all) preservation-related data to enable future access. The process is so difficult that it has been described as requiring “heroic measures” [72]. 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. Victoria Electronic Records System (VERS) Encapsulated Objects are geared specifically to Australian government records [145] 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 [40]. The Metadata Encoding and Transmission Schema (METS) only requires a file inventory and structural map, with metadata like rights and provenance completely optional [88]. 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 [148]. Just cataloging the underlying resource type is difficult, as we will see in Chapter VII on page 84, 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

TABLE 3: File & Structural Map Sections of METS

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

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 [114] 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 [96, 100]. 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 & PREMIS

The METS model is at the heart of many repository systems, including the widely used software packages DSpace and Fedora. The primary object, a METS “document,” contains seven major sections (Figure 5 on the following page), but only the File and the Structural Map sections are required (see Table 3). 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 Core and MARC are recommended. For the Administrative section, recommended schemas include

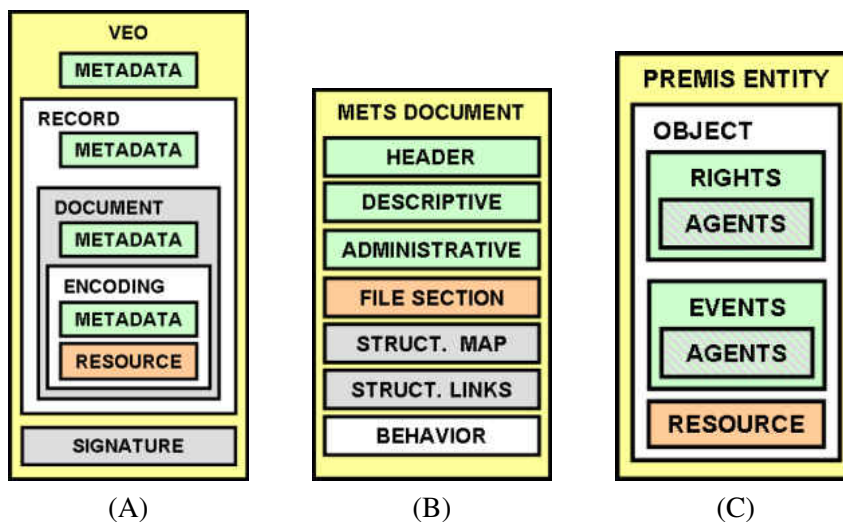


FIG. 5: (A) The VEO, a VERS Encapsulated Object; (B) The METS Document Object and (C) the PREMIS Intellectual Entity. Each is a different implementation of the Complex Object type.

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 on the following page), 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 can be derived for our sample resources.

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 web site, 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 web site could be mapped to an “Intellectual Entity” with each of the resources comprising an “Object” contained within that entity. Figure 5–(B) 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 ingestion can be when two sources implement a model like METS in different ways [97]. 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 [143]. VERS metadata objects are designed to ensure authenticity. The VERS workflow (shown in Figure 6 on the following page) 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 [139]. 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 web site is feasible [124].

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 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 7 on page 18 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

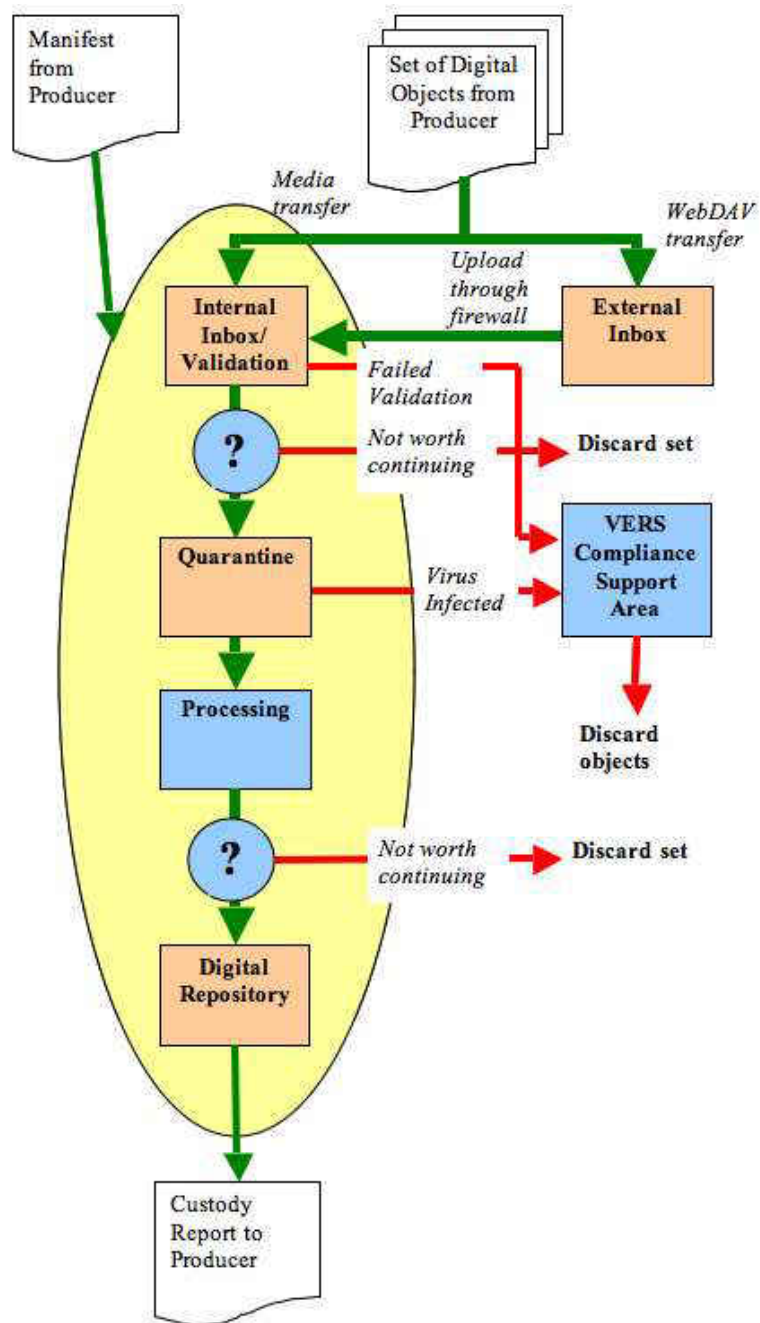


FIG. 6: The VERS Workflow, as described and pictured in [144].

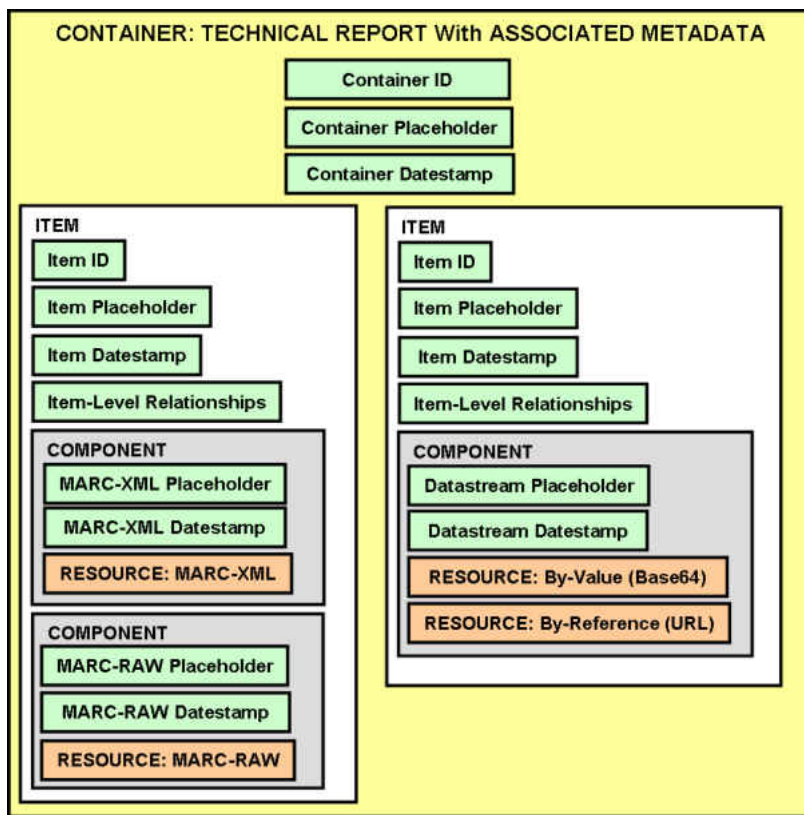


FIG. 7: The LANL MPEG-21 DID Complex Object

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 C on page 164– 1 on page 164). The Jhove metadata would be contained within one *item-component* in the container, and the Acroread information (Figure 8 on the following page) 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. Harvesting a web site 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 [82]. Since very long term availability of resources is not guaranteed by digital library 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. 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

```

<?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"?>
<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-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. 8: Metadata derived from Foo.pdf using Acroread.

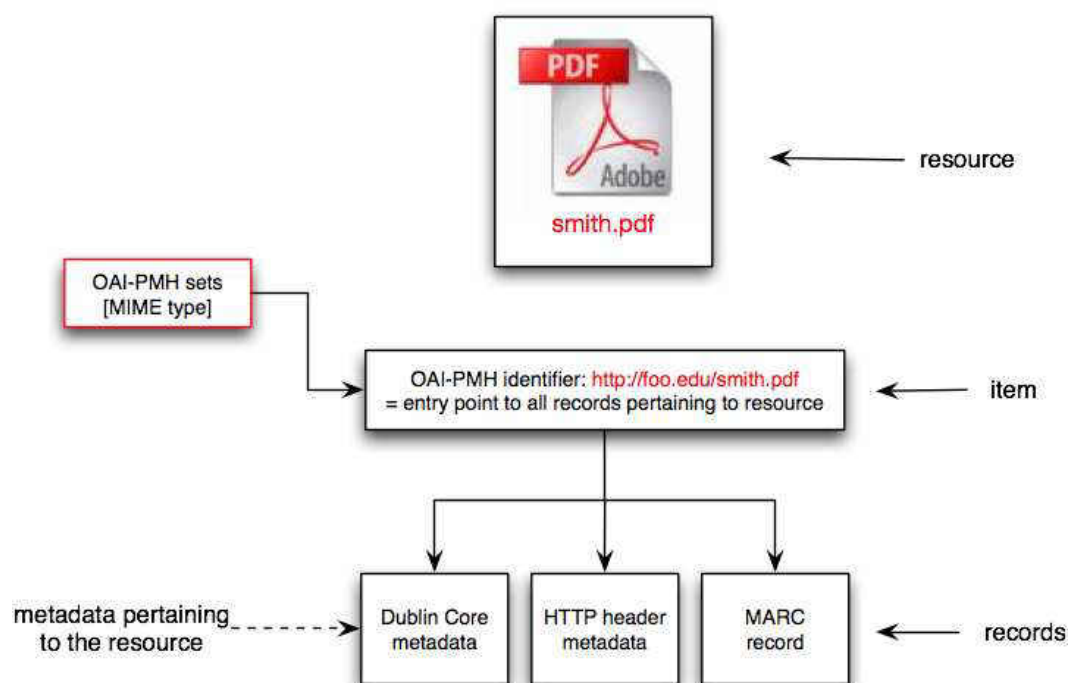


FIG. 9: The basic OAI-PMH Data Model. This view is adapted from the model presented in [136].

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 9. 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 timestamp 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 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.

TABLE 4: OAI-PMH Verbs

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)

OAI-PMH supports six verbs or “protocol requests” which are listed in Table 4. 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 types (e.g., image, audio) and subject area (e.g., USHistory, anim ) 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 since 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 harvester (or service provider) is an application that issues OAI-PMH protocol requests in order to harvest XML formatted metadata. In ReST terms (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 [69]. 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. The fundamental, distinguishing characteristic that separates harvesting with OAI-PMH from regular web crawling is that the repository chooses the size of the *resumptionToken*, not the harvester. This allows repositories to dynamically throttle the load placed on them by harvesters. The format of the *resumptionToken* is not specified in the protocol and is left to individual repositories to define. Load-balancing, throttling and different strategies for *resumptionToken* implementation are discussed in the OAI-PMH Implementation Guidelines [69].

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 [148], 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 - *resource, metadata, records, and items*. In some cases, it means using OAI-PMH for other than typical bibliographic scenarios [137]. 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. 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 10 on the next page.

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 [149, 45, 36]. It is an HTTP-based protocol designed to allow incremental harvesting of XML metadata [135], 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.

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 [76]. 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

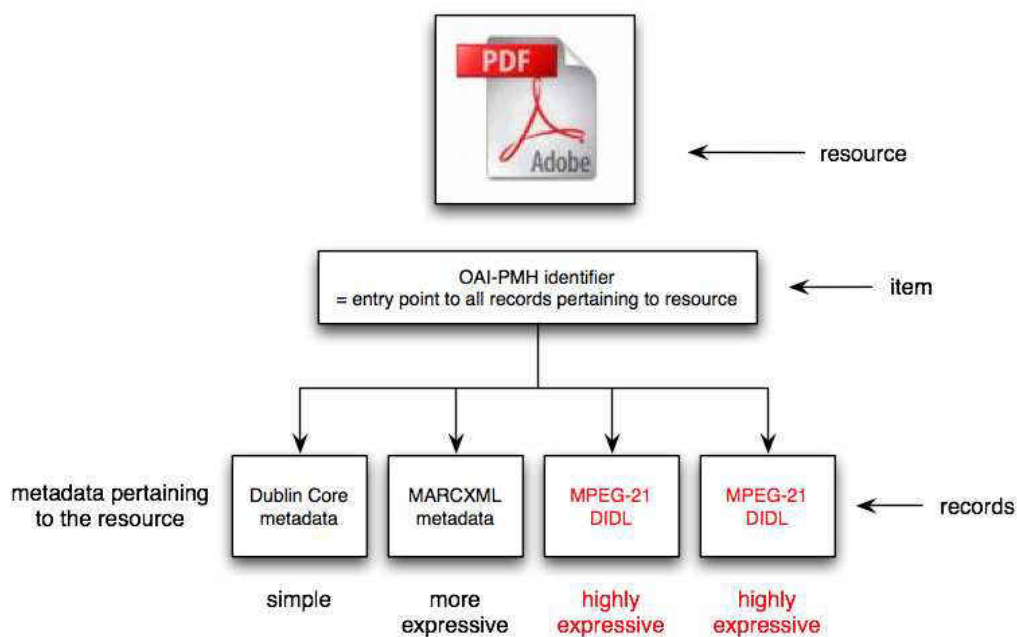


FIG. 10: OAI-PMH can include complex objects as metadata. The MPEG-21 DIDL and METS are complex object types. (Image adapted from [136]).

authorization keys are no longer maintained now that the company is out of business [32]. Law suits caused the company’s demise (legal aspect), the software can only be activated by a key found on the original web site (technical aspect), and the legal problems arose because commercial DVD producers believed they were losing money to users of DVDCOPY (economic aspect). For the original DVDCOPY 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 [125]. 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 [125, pages 279–292]. A similar dilemma was recently raised with the publication of nuclear-bomb fabrication information from an archived copy of an Iraqi web site, restored as part of a US government initiative [17]. Another example which featured prominently in recent headlines is the web site of a congressman accused of soliciting (male) Congressional pages [49].

While financial motives may be an obvious factor in the “preservability” of a web site, 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 WWW 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 web sites 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 democratization 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

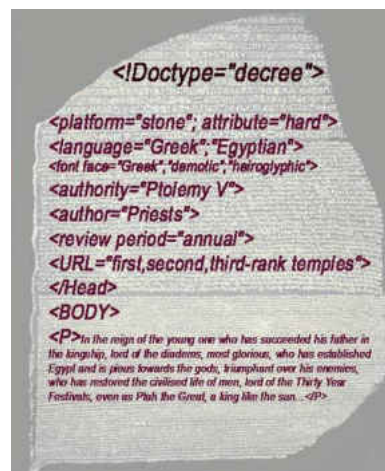
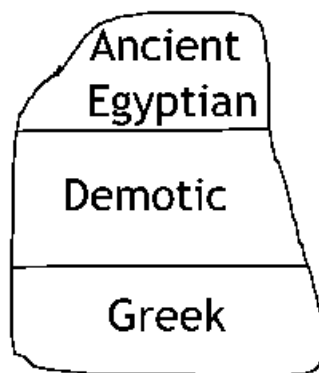
1 INCIDENTAL WEB PRESERVATION: USENET

In 1997, Hauben and Hauben [54] 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. This “net” had been already available for two decades at that point (since 1979), but it pales in comparison to the Web we have today. Once *the Net* became *the Web*, many of the servers that had provided *BBS* (Bulletin Board) services went dark, i.e., they were no longer available. With them went many notable Net items, like the announcement of the World Wide Web from Tim Berners-Lee [44] and the first posting mentioning “MS-DOS.” [43]

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 [42]. What makes this remarkable is that these files were not recovered from a preservation archive but from the archived copies made by *everyday users* when Usenet was in its prime. In some cases, the data came from the Exabyte and DAT tape archives 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).

There were many factors contributing to the success of the Usenet project. One was that the timeline was relatively short and 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 [113]. ASCII or an extended variant of ASCII is still used worldwide. Just as Greek was the deciphering key provided on the Rosetta Stone (Figure 11 on the following page), ASCII was the key to deciphering the bits on the various media.

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.



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

FIG. 11: 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 [132]; (B) is from [34]

2 INTENTIONAL WEB PRESERVATION: 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 web sites [146], not just once but several times over the course of years. The IA hosts a site called the *Wayback Machine*¹ 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 12 on the next page). 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.

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 [18, 24]. Getting a complete listing of *possible* and *accessible* URLs at a site is no easy task [16, 55]. 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.

There are often long delays between Internet Archive crawls and the posting of a site snapshot,

¹<http://www.archive.org/web/web.php>

TABLE 5: 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/

preservation, in part because it plays a backup role rather than a historical archive role [12]. 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.

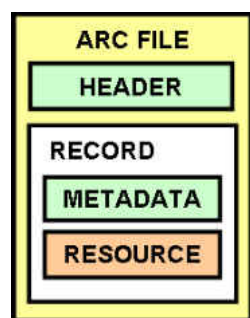
The Internet Archive stores crawled sites in a file format called *ARC* [74], shown in Figure 14 on page 31-(A), with a command-line example shown in Figure 14 on page 31-(B). Except for the protocol headers, web crawling using HTTP, FTP, and NNTP typically generates little or no explicit descriptive metadata. A web site merely needs to be crawled by the Alexa robot for the ARC file to be created, or it could use IA’s Heritrix [91] 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*. [2] 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 13 on the next page. 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?

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

<pre> <?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>2007-02-03 T18:22:23-05:00</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> <profiles><profile>JFIF</profile> </profiles> <properties> <property> <name>JPEGMetadata</name> <values arity="List" type="Property"> <property> <name>CompressionType</name> <values arity="Scalar" type="String"> <value>Huffman coding, Baseline DCT</value> </values> </property> <property> <name>Images</name> <values arity="List" type="Property"> <property> <name>Number</name> <values arity="Scalar" type="Integer"> <value>1</value> </values> </property> <property> <name>Image</name> <values arity="List" type="Property"> <property> <name>NisoImageMetadata</name> </pre>	<pre> <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/ 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:ImageCreation> </mix:ImageCreation> <mix:ImagingPerformanceAssessment> <mix:SpatialMetrics> <mix:SamplingFrequencyUnit>3 </mix:SamplingFrequencyUnit> <mix:XSamplingFrequency>0 </mix:XSamplingFrequency> <mix:YSamplingFrequency>0 </mix:YSamplingFrequency> <mix:ImageWidth>459</mix:ImageWidth> <mix:ImageLength>253</mix:ImageLength> </mix:SpatialMetrics>boxedminipage <mix:Energetics> <mix:BitsPerSample> 8,8,8</mix:BitsPerSample> <mix:SamplesPerPixel> 3</mix:SamplesPerPixel> </mix:Energetics> </mix:ImagingPerformanceAssessment> </mix:mix> </value> </values> </property> <property> </repInfo> </jhove> </pre>
--	---

FIG. 13: Part of the metadata derived from Barfoo.jpeg using Jhove's JPEG-HUL module. See Appendix C on page 164 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. 14: 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>).

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 14.

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.

3 OTHER WEB ARCHIVING EFFORTS

Web site preservation is the mission of organizations like Japan’s National Diet Library [142] and the National Library of Australia Digital Services Project. [94] The U.S. has funded the National

Digital Information Infrastructure Preservation Program (NDIIPP) in an effort to preserve as much of the web as possible [95]. The mission of the Henry A. Murray archive at Harvard University [52] is to “preserve in perpetuity all types of data of interest to the research community”, much of which is now web based. Both Holland [66] and the United Kingdom [93] 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.” [33] The primary focus at this point is European heritage, including non-web-based materials. Despite these efforts, as noted in Chapter II on page 9, 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.

4 THE WEB SERVER AS AGENT OF MIGRATION

Fashions come and go in digital file format 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. These features, plus the Kindle’s form-factor have resulted in the creation of yet another file type specific to the Kindle. Will this become the next “PDF” – that is, will documents transition to an interactive format used by e-book readers? This format is very 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 over 100,000 book titles with more being added daily. Despite its relative newness, the new format is gaining rapid acceptance.

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. Amazon states that the 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. Nonetheless, this conversion/migration is an intriguing example of a nearly interactive migration process for digital documents arising directly from web services.

On the web itself, HTML appears to be giving way to XHTML and SHTML along with Javascript and other dynamic content generation techniques. [35, 78] Throughout these changes, both web servers and browsers have aided migration by the use of MIME-typing to define the *kind*

of file sent in the response. Consider the example in Figure 15. Browsers, the main users of web content, have also acted as agents of migration by having 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. In other cases, interest in incompatible types, such as the Virtual Model file type of Figure 15 may generate tools to convert the content into something understandable, thus enabling content types both new and old to be accessible to users. The author's recently installed 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. Again, the democratization of web services and tools has improved the likelihood of digital content preservation.

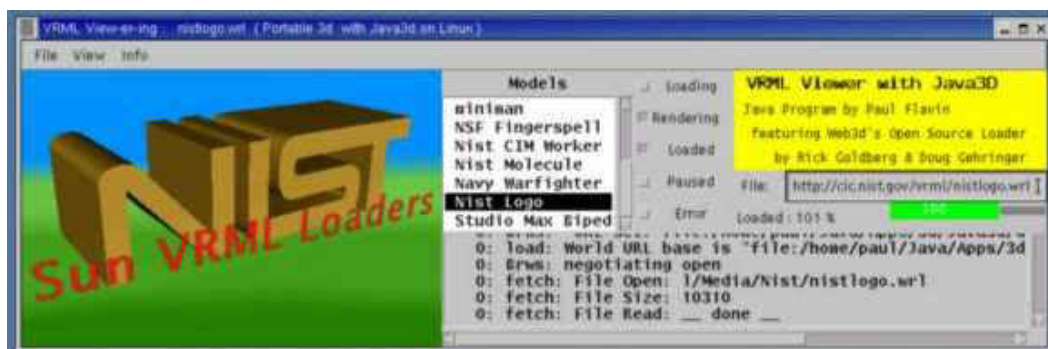


FIG. 15: Unhandled MIME Type & Resource Migration. On the top is the error message generated by the author's web browser when attempting to get the original VRML file, *nistlogo.wrl*. The web server knows the MIME type, but the browser has no plugin to handle it. It defaults to a decision box asking what to do with the resource. On the bottom is another web server [39] where the file has been converted into something currently understandable by the browser. The received MIME type in this case is image/jpeg.

5 SUMMARY

The benefits of “preservation by popular demand” have already been seen in the Usenet restoration undertaken by Google. Internet Archive’s efforts have proved important to many people, and have helped restore websites around the globe [87]. 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 archiver. In the end, grass-roots preservation efforts and popular personal archive collections may be what provides the future with insight into this generation’s digital heritage.

CHAPTER IV

THE CURRENT ROLE OF SEARCH ENGINES IN DIGITAL PRESERVATION

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 [16, 120, 85]. 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 [105]. 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, [48] 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 [55, 24, 27], or on improving site performance and accessibility to crawlers [16, 110]. 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.

1.2 Crawler Behavior on Websites with Disappearing Content

The “Sliding Directories” experiments 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 for the index.html files in each bin, which were modified when links to deleted web pages were 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:

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

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

$$Total_b(b, d) = HTML(b, d) + PDF(b, d) + IMG(b, d) \quad (2)$$

The total number of resources in each update bin decreases with the bin's periodicity as show in Figure 16. The number of HTML, PDF and image files in each bin b on any day d is defined as:

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

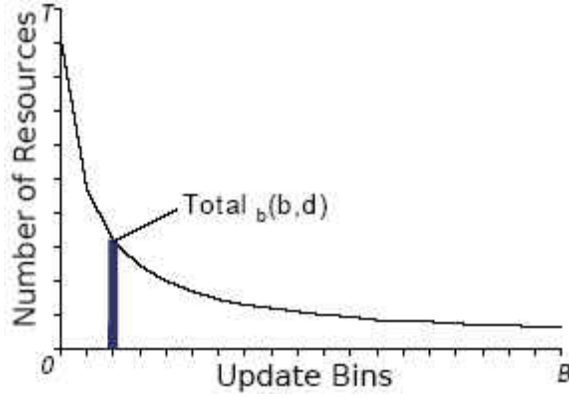


FIG. 16: Number of resources in the test website

$$PDF(b, d) = \lfloor T/b \rfloor - \lfloor d/b \rfloor \quad (4)$$

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

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 & 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 17 on the following page illustrates the website structure; Table 6 on the next page 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 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

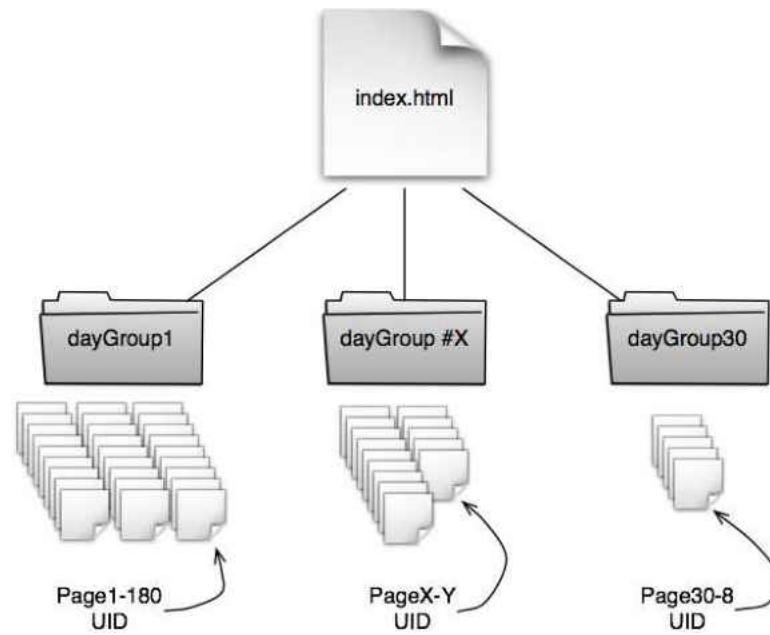


FIG. 17: The number of resources in each subdirectory varied. In subdirectory 1, a resource was deleted every day; in subdirectory 15, a resource was deleted every 15 days, etc. By the end of the 90-day experiment, all of the directories are empty.

TABLE 6: Website content of the Sliding Directories experiment

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 :

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

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. To accomplish resource management and accurate update, the author wrote a script to automatically remove one or more resources from the site every day. The total content of the directories thus “slides” downhill over the course of the experiment, as shown in Figure 18 on the following page. 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 directory set a “subsite” of the host website [140]). 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.

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 begin 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, time stamp of the request, the request itself (which contains the HTTP verb, 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 in Table 7 on page 41. 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

FIG. 18: Resources are removed gradually over a 90-day period. Green represents “live” resources, while red indicates those that have been removed. Note the downhill sliding pattern as the experiment day number increases. The crawl data is mapped on top of the disappearing resource picture (MLN Website) as blue “snowflakes”. An interesting animation of the crawl while the site disappears can be seen in [128].

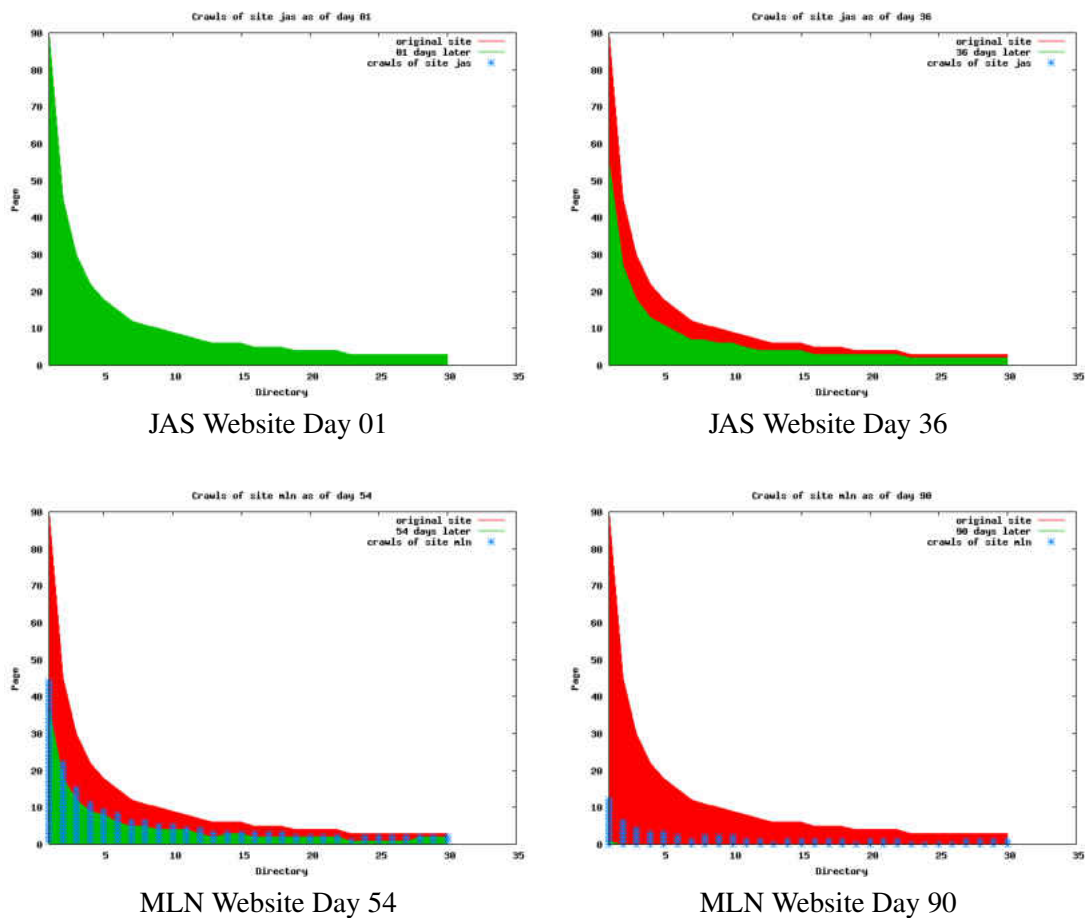


TABLE 7: Log data from two of the Sliding Directories sites clearly show how the amount of information available from a web server log can vary. Dash in the column indicates the field is tracked but empty for this record. For an explanation of the fields, see [133].

Log Field	OBR Site Log Entry	JAS Site Log Entry
host (IP)	access.log.25:66.249.66.69	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	<i>not used</i>
uagent	"Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"	<i>not used</i>

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 errors 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 that arises when visitors are not identified (the uagent field is not tracked) is determining authoritatively who owns the visitor IP address. 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 entries of record as the final arbiter.

Other identification problems 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. The author treated Inktomi robots as Yahoo robots. Google, Inktomi/Yahoo, and MSN comprise the bulk of log records (nearly 80%) for non-spam search engines. Traffic was minimal from other popular engines like “Picsearch” and “Internet Archive”, with only 157 total requests from Internet Archive; 315 from Picsearch. Table 8 on the following page summarizes the crawls by search engine.

TABLE 8: Crawler Statistics from the Sliding Directories experiment.

Crawler	Total Requests by Site			
	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, identify unique remote host names per 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 [128]. A summary view of the coverage for the MLN site is shown in Figure 19. 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 20 on page 44.

Experiment Results

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 [128] for a dynamic view of the behavior). These resources also persisted longer in cache [87], 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

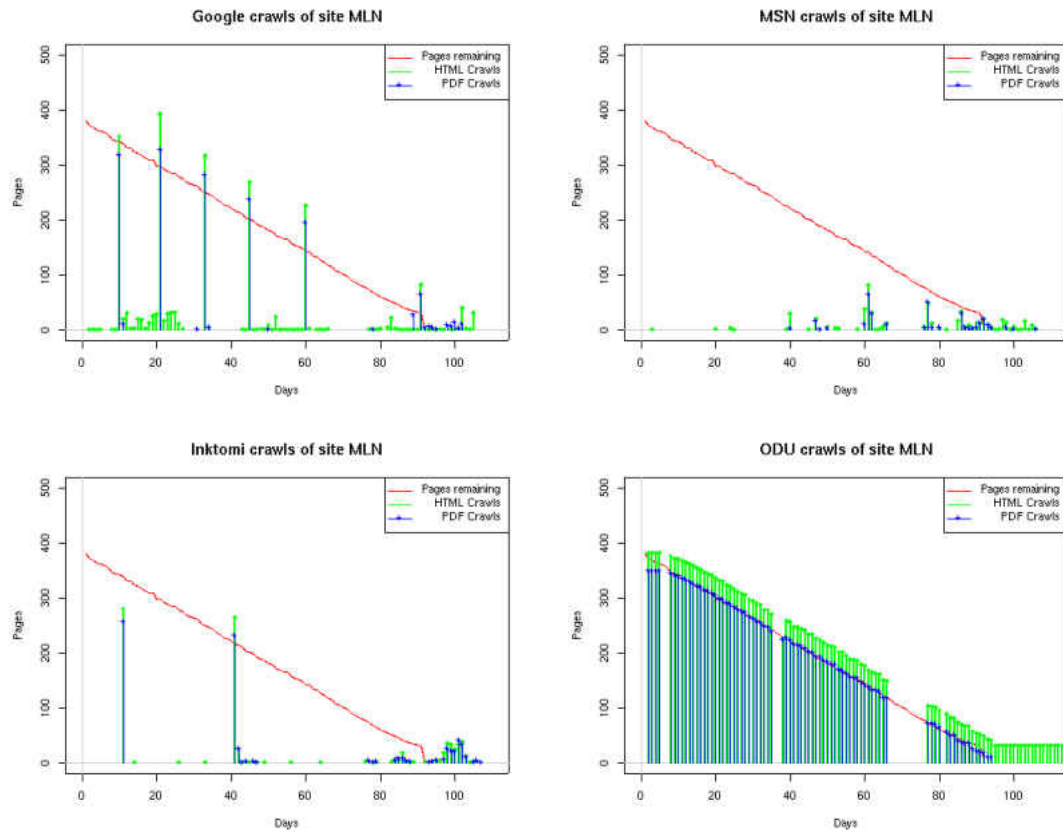


FIG. 19: Crawling patterns of Google, MSN, Inktomi/Yahoo & the ODU Crawlers on site MLN. The ODU robot was the heaviest user of the site, as the graph shows. The red line indicates the original limit of source content. Robots sometimes attempted to access (guess?) URLs that did not exist.

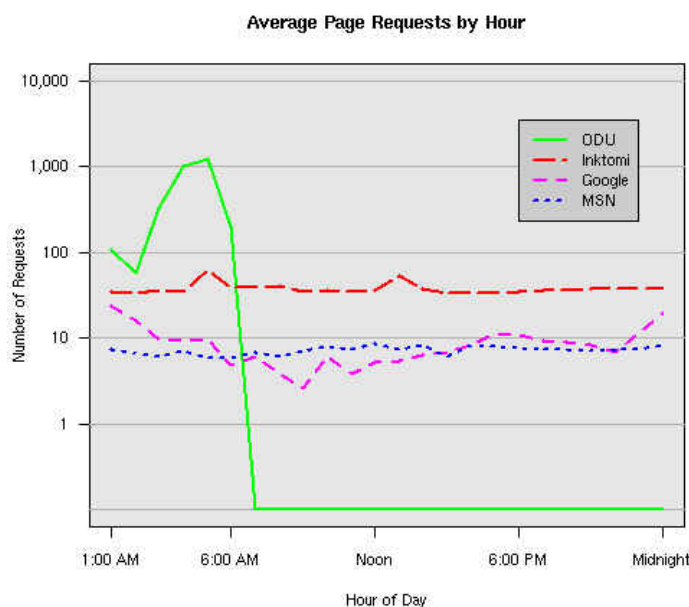


FIG. 20: Crawling patterns by time of day for Google, MSN, Inktomi/Yahoo & the ODU Crawlers on site MLN. ODU’s robot limited its crawls to the early morning hours, whereas the major search engines maintained a relatively steady pace of requests throughout the day.

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.

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. At only 3 levels, the Sliding Directories experiment of Section 1.2 on page 36 in this chapter did not have sufficient depth to test the willingness of crawlers to explore beyond the topmost levels of a website. 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. The author created a series of live websites, two dot-com sites and two dot-edu sites, that 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 & Implementation

Reviewers of the Sliding Directories experiment wondered whether the artificial content of those sites influenced crawler behavior. Given the persistent behavior of crawlers on the Sliding 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 ¹ 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: 100 groups of 100 directories, each containing 2 pages (100 x 100 x 2 = 20,000), plus index pages for the various directories, and an assortment of images. Figure 21 on the next page shows the structure of the website.

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 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 22.

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:

- <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)

¹<http://www.gutenberg.org>

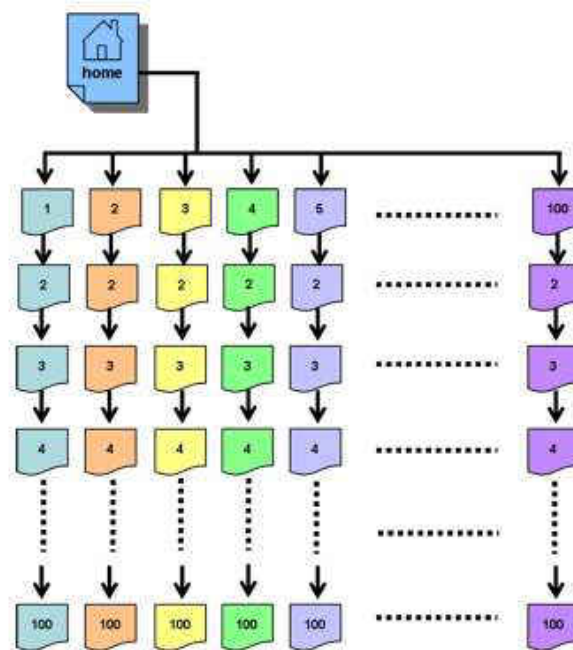


FIG. 21: Both Bread Crumb and Buffet sites were very deep as far as directory structure depth. The main difference was in the location of links to each of the site's pages.

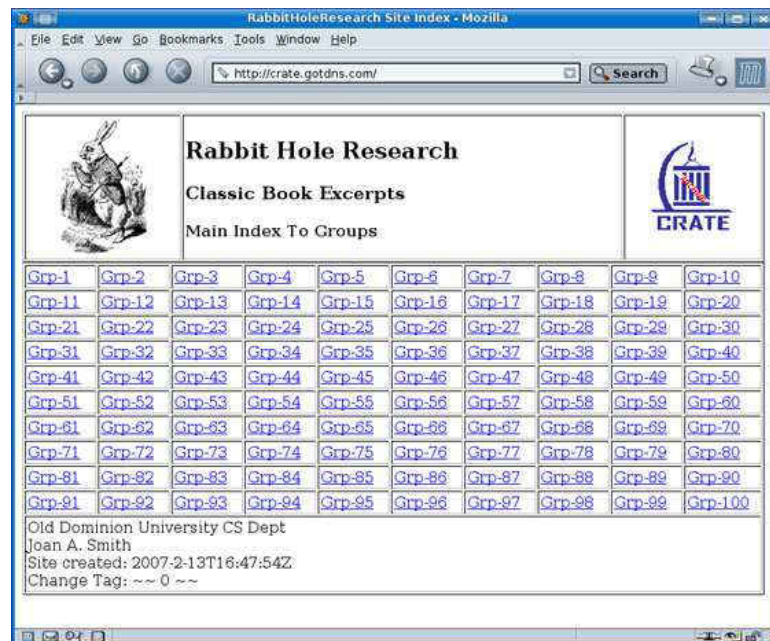


FIG. 22: The entry page to the test website linked to the top page in each of the 100 subdirectories.

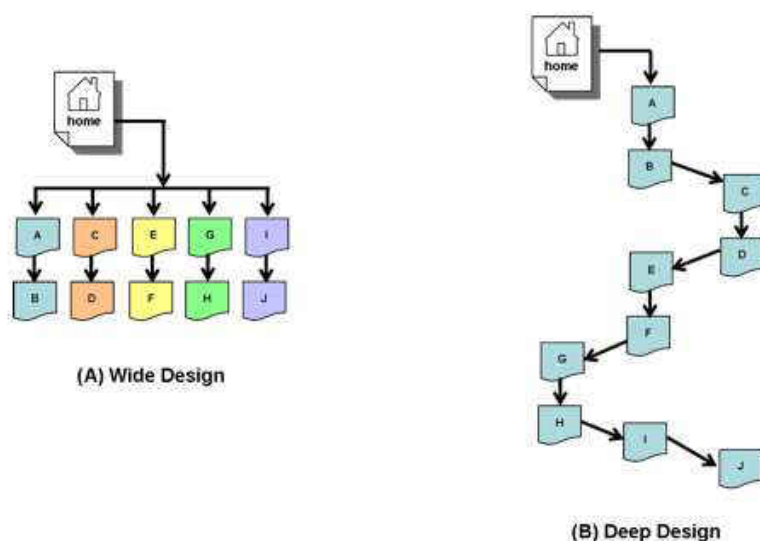


FIG. 23: Wide & Deep Sites differ by the “size” of the links, as counted by the number of slashes in the URI. The path to resource J in (A) would be `http://foo.edu/I/J` whereas the link to J in (B) would be `http://foo.edu/A/B/C/D/E/F/G/H/I/J` – a deep link. For Buffet sites, a list of each link was provided near the top of the website, effectively giving a short-cut route to the deepest point. Bread Crumb sites had only one link per page and so the full depth had to be explored one link at a time before reaching the bottom.

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. Figure 23 illustrates the difference between these two design approaches.

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

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

The HTML structure of each site was very similar, except for the subdirectory linking method. Buffet Sites had a high-level index, as shown in Figure 24-(A). The Bread Crumb sites provided only one link per page as shown in Figure 24-(B). Since search engines harvest content rather than design, and since pre-designed web templates are replicated all over the web, it seemed extremely unlikely that common design would have any effect on search engine crawler patterns at the sites. Each site has about 15 MB of text content, and a few hundred Kilobytes of images culled from

public domain sources. There is no hidden content, and since the text was extracted from classic literature, it "makes sense" grammatically.

Data Collection

All four sites were installed simultaneously in February 2007. From that point forward, the logs were harvested daily for crawler activity by Google, Yahoo, and MSN (only). The author's utilities from the Sliding Directories experiment were adapted to collect data from the Deep Website experiment and map 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 dot-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 dot-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.

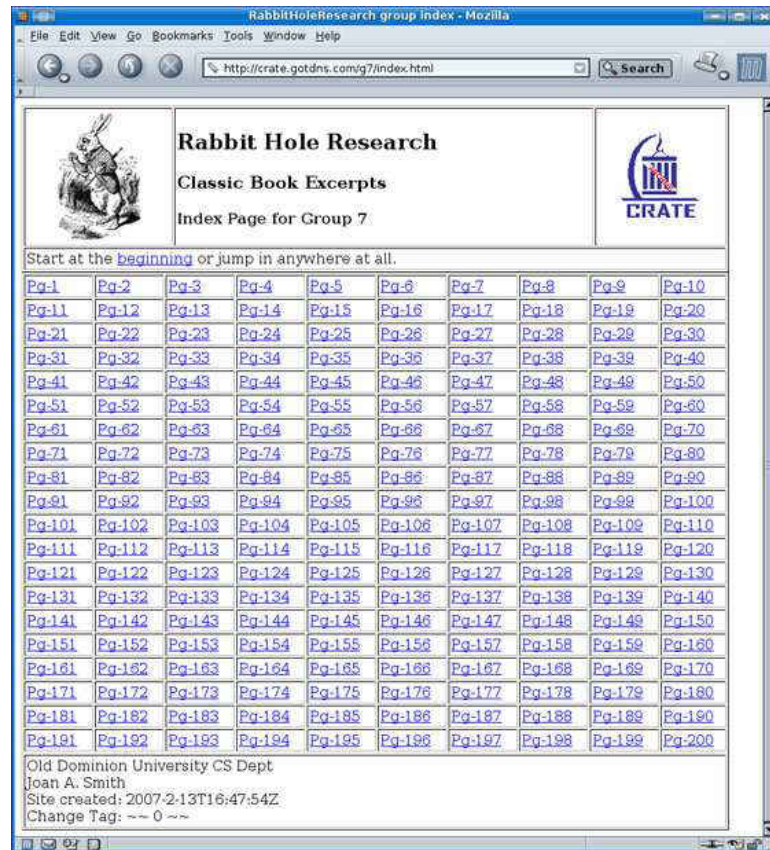
Crawler Characteristics

Each crawler exhibited different access and persistence patterns, and these patterns varied by domain (dot-com or dot-edu). In general, width was crawled more thoroughly and quickly than depth. Upper-level "index" 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.

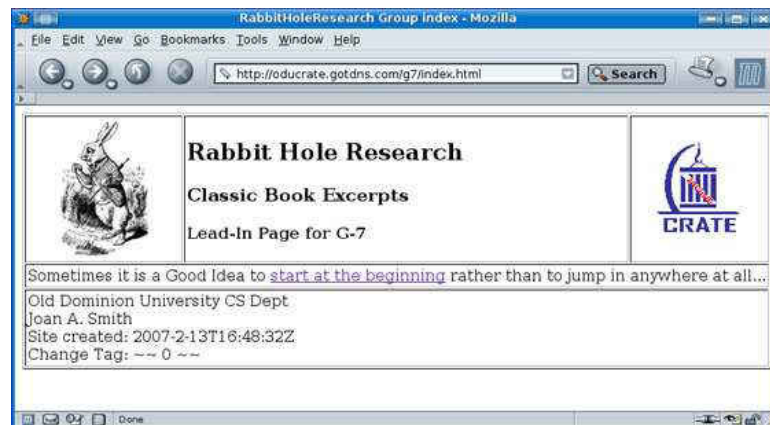
The logs show some interesting site access patterns by the Google, Yahoo and MSN crawlers. Figure 25 is a series of snapshots of the Googlebot's progress through the Bread Crumb site, "oducrate.gotdns.com". These are best seen in an animation of the activity, available in [131], where Google's robots advance through the Bread Crumb dot-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 was considerably more random, as Figure 26 shows.

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 27 on page 53 and 28 on page 54. 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" [106] certainly exist in



(A) Buffet site subdirectories had an index page linking to each resource below.



(B) Bread Crumb site pages linked only to one resource at a time.

FIG. 24: Examples of the entry page to subdirectories for the Buffet and Bread Crumb sites

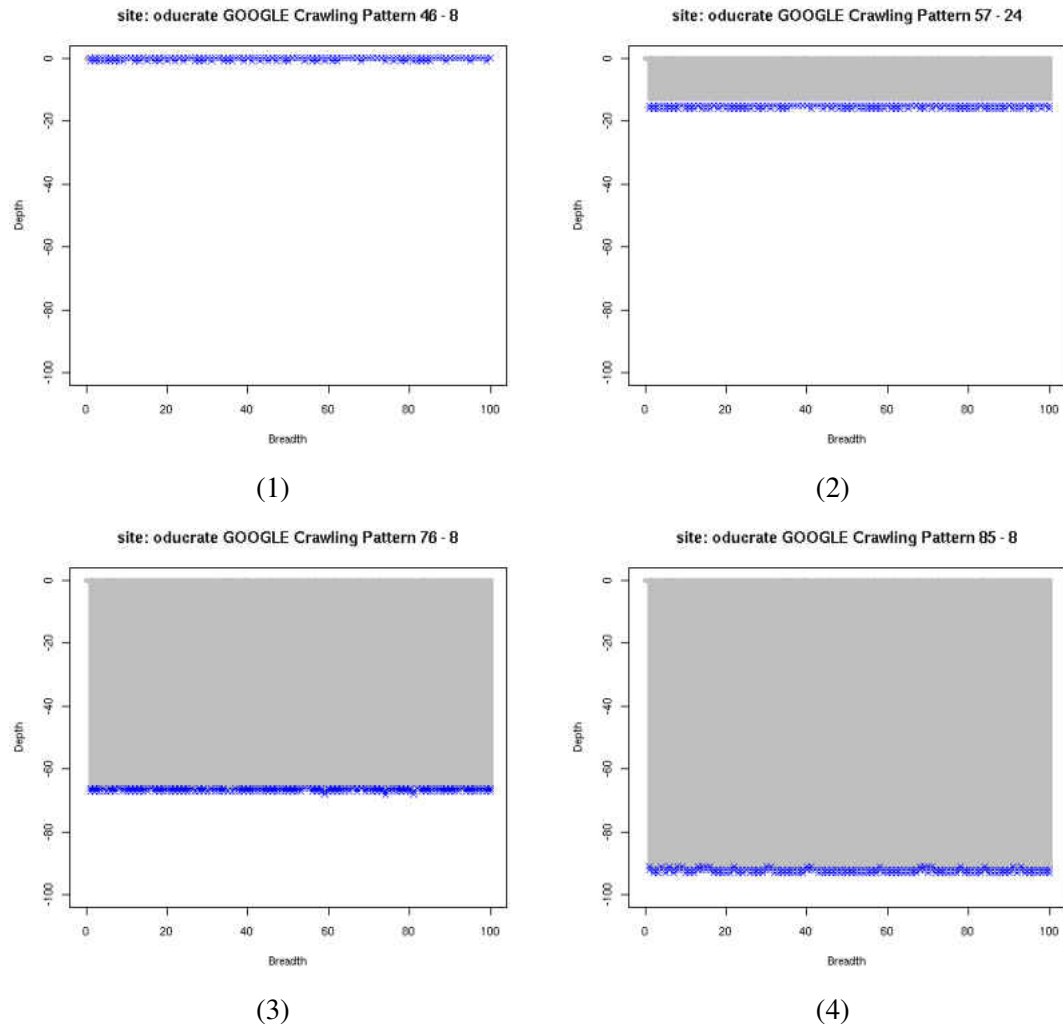


FIG. 25: Crawling patterns of Googlebot on the Bread Crumb site (oducrate.gotdns.com). Note the almost regimented progress through the links. Gray color indicates links that have been crawled on a prior day, and blue indicates the active request of the moment. Google and the other search engines have numerous individual robots that may visit a site simultaneously to harvest the data.

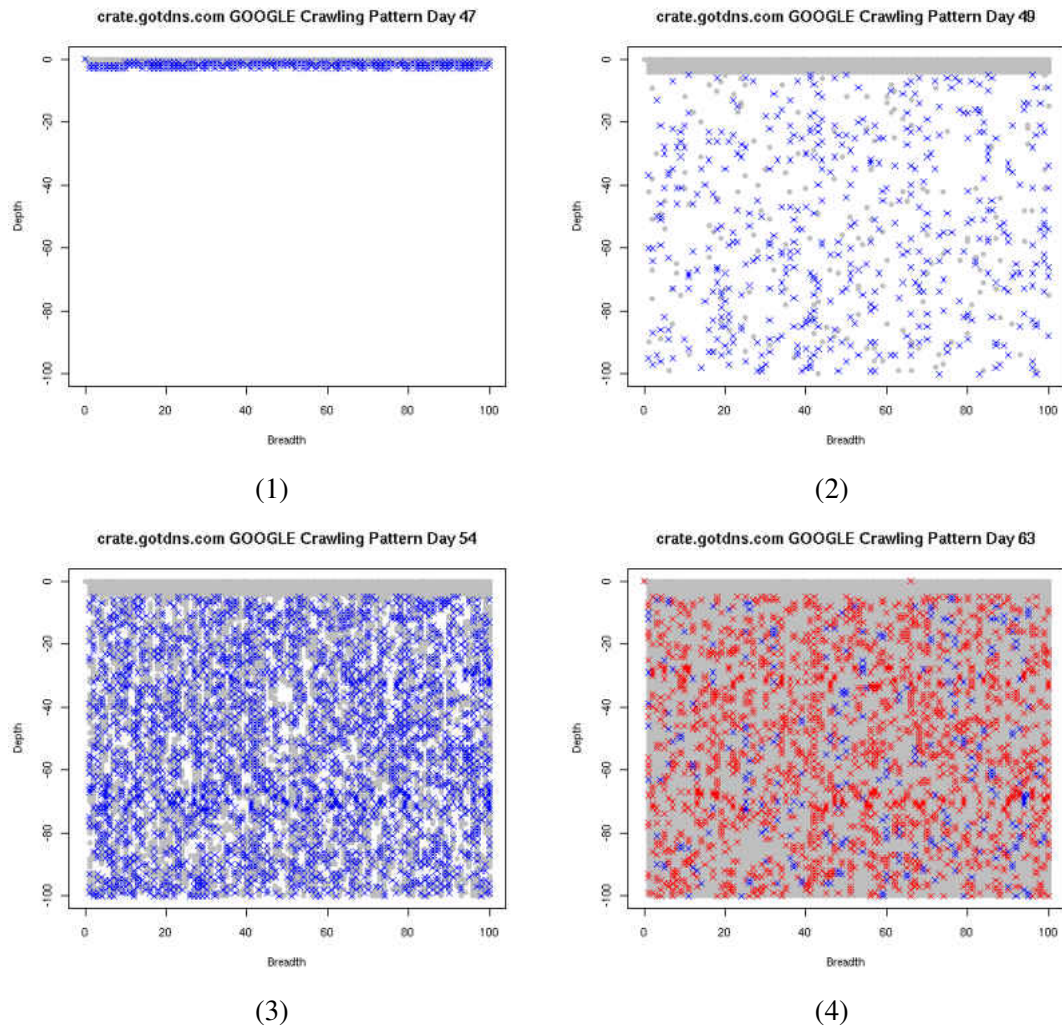


FIG. 26: Googlebot crawling patterns on the Buffet site (crate.gotdns.com) are very different from the Bread Crumb site. Here, the robots take advantage of the top-level list of resources and simultaneously harvest pages scattered throughout the site. Google also uses its request patterns judiciously. Red indicates a Conditional Get, blue is a regular get, and gray is a page that has been visited on a prior day.

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 Sliding Directories experiment), many pages on the sites would not have been discovered at all.

1.4 Summary of Crawler Observations

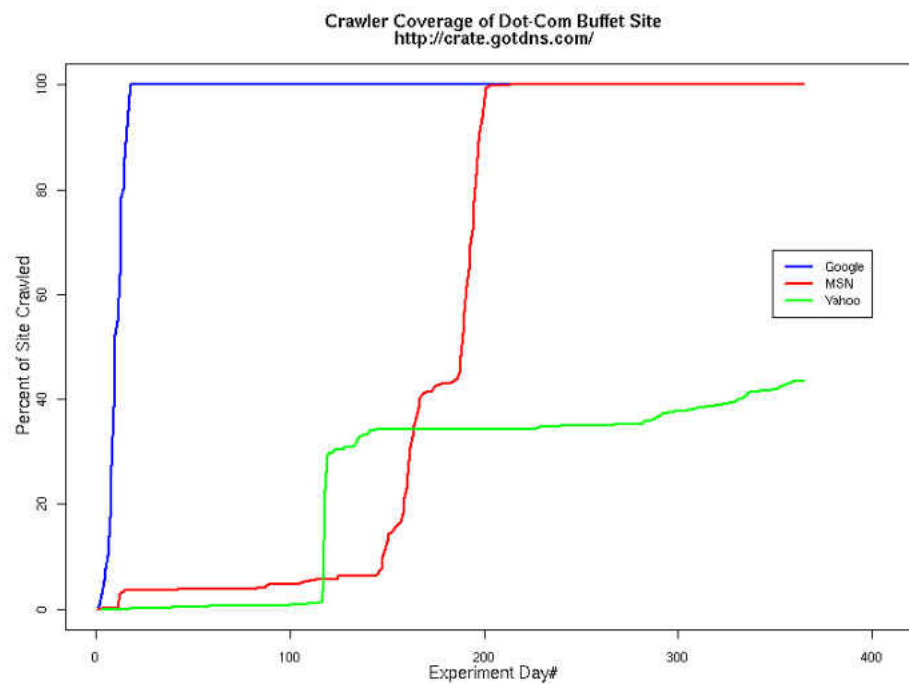
Crawlers exhibit a preference for “wide” rather than “deep” sites, and for sites which provide high-level indexing of site contents. Some search engines are more thorough than others for sites that do not meet this criteria, but preservation is best served by maximum exposure of site resources. Cooperating 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

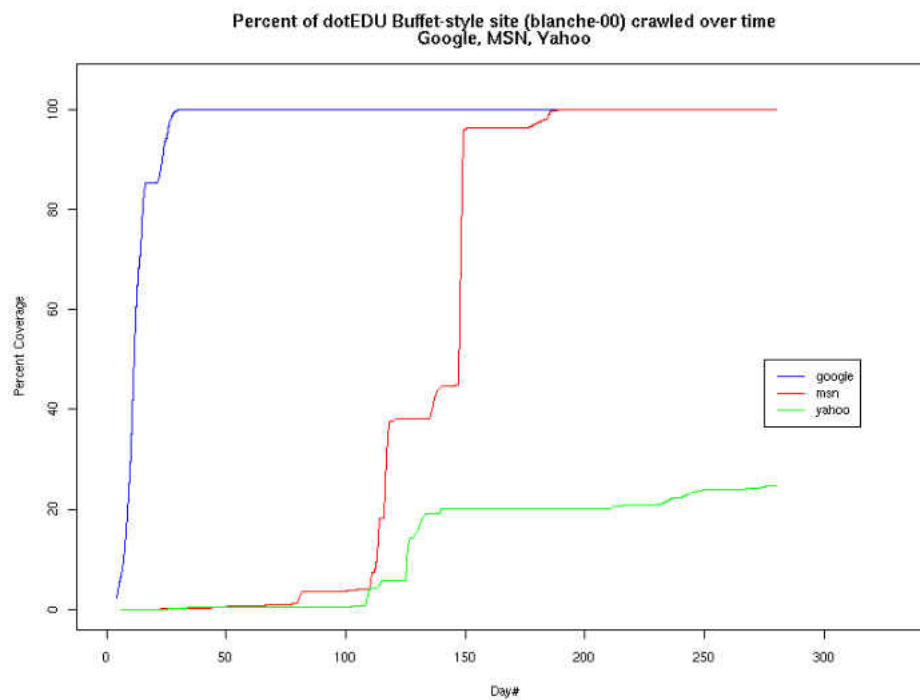
An important preservation task is *refreshing* the bits. Search engines assist with this task through their continuous crawls of websites, which are then accessible to the general public. A user could, at any point, store a copy of all or part of a website thus acting as an incidental preservation agent. 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 [128] and [131] show an apparent change in the access frequency of crawlers between the first Sliding 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 is failing to refresh the bits, instead relying on its cache in the expectation that the cached copy is still intact. MSN and Yahoo are refreshing their copies of the resource with each visit.

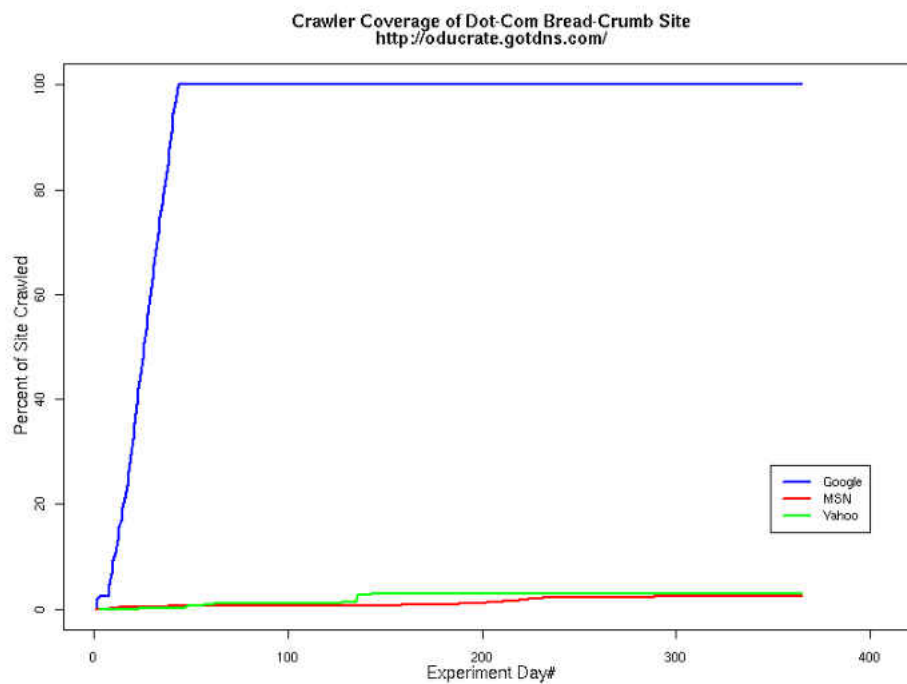


(A) Dot-Com Domain Buffet Site

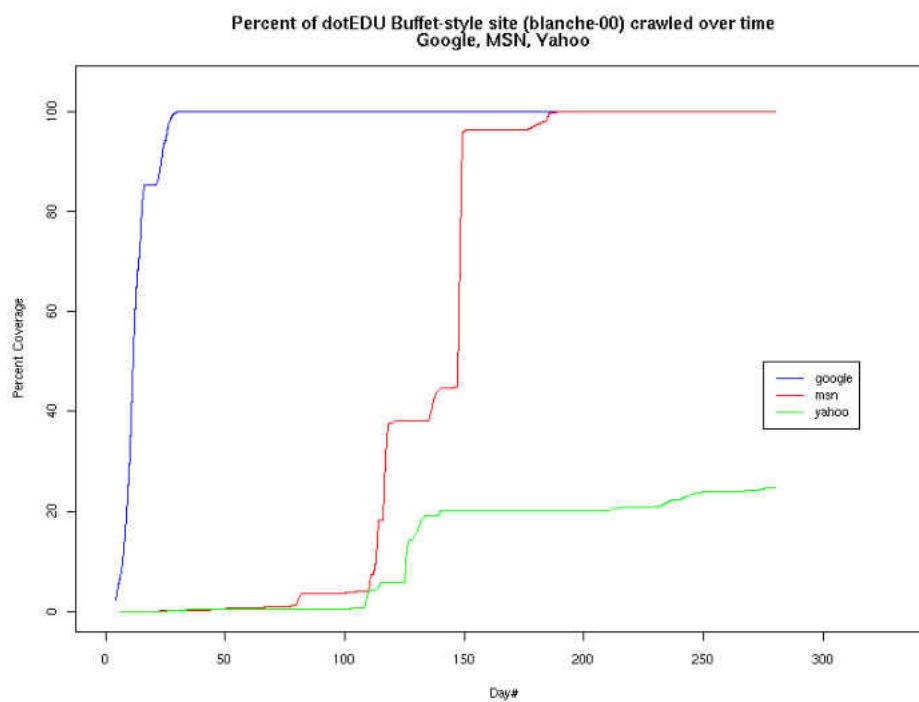


(B) Dot-Edu Domain Buffet Site

FIG. 27: Buffet Site crawling pattern by the Big Three search engines. 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 “map” of the site.



(A) Dot-Com Domain Bread Crumb Site



(B) Dot-Edu Domain Bread Crumb Site

FIG. 28: Bread Crumb site crawling pattern. Although all of the site's links are accessible, *every page* must be individually crawled to find them all. The crawl patterns here differ considerably from those in Figure 27 on the preceding page, implying that site design does affect crawlers.

3 THE SEARCH ENGINE AS AGENT OF PRESERVATION

3.1 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.

3.2 Lazy Preservation

The term *Lazy Preservation* was coined by Michael Nelson, and adopted by Frank McCown and the author [98, 87] to describe the ability to recover websites from the web infrastructure (WI). In some cases, a complete website could be reconstructed using only the content from search engine caches [65]. A set of tools developed by McCown [83, 84] has been used by numerous sites worldwide to restore lost websites. The WI 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

1 THE COUNTING PROBLEM DEFINED

Websites consist of resources which may be files, dynamic content, or a combination of different types of content. The collection of resources is what defines the site:

$$W = \{w_1, w_2, \dots, w_n\} \quad (7)$$

This is true whether or not the members of W are discreet entities on a file system, and whether or not the file system contains entities that are not part of the website. Regardless of the mapping from w to virtual or actual resource, website preservation expects a list of each $w \in W$ so that all accessible resources which make up web site 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.

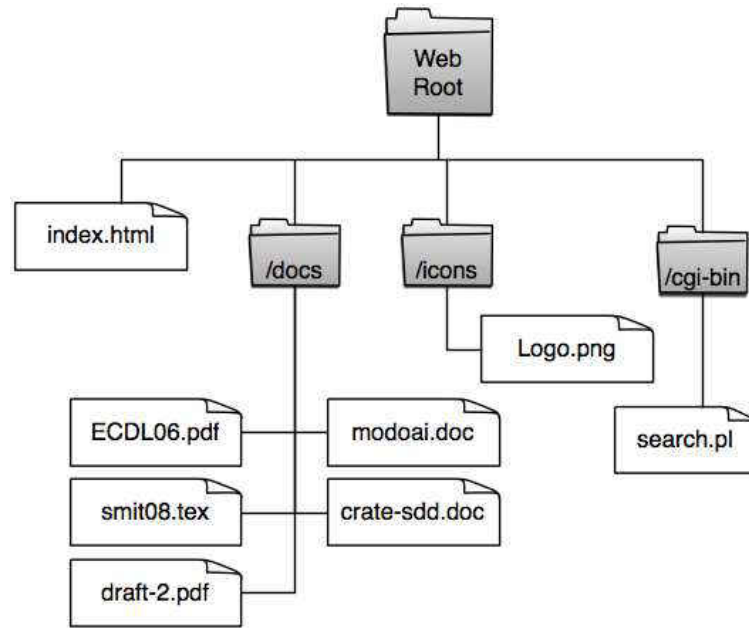
2 WHY THE COUNTING PROBLEM EXISTS

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.

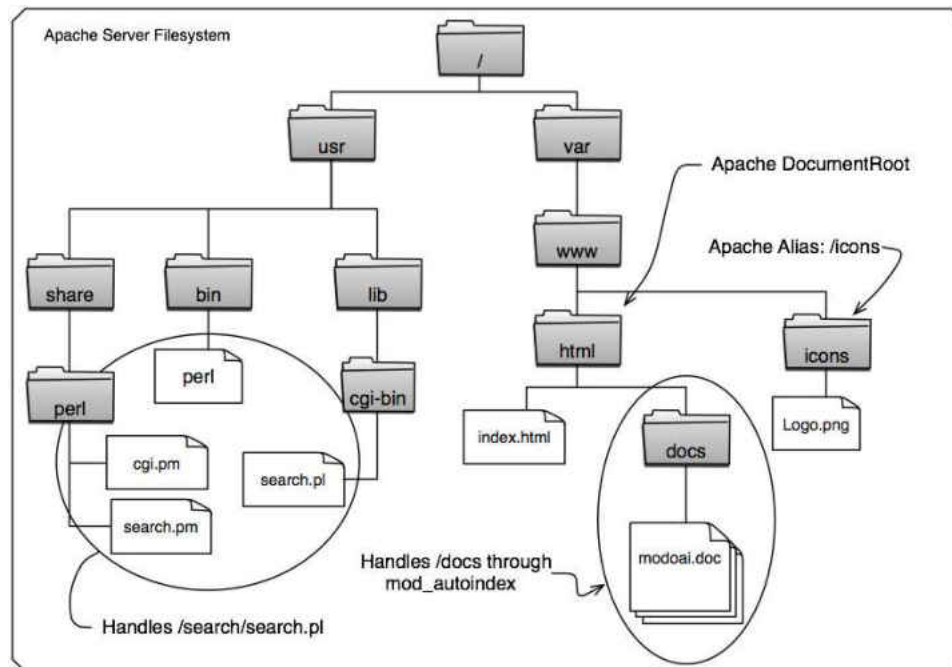
One reason the problem exists is that website file space does not always map directly to website URLs. Figure 29 on the next page shows how different parts of a website intersect with the website’s host file system. Another reason that the problem exists lies with the underlying protocol, HTTP, and the limitations of crawling related to that. These two issues 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,



(A) Website Map



(B) File System Map to Website

FIG. 29: 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.

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’s methods do not have SQL-like syntax. The protocol is designed to request 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.

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 [50], and the “deep web” is estimated to be 550 times larger [11]. 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 [102, 25] and to improve crawling strategies [26, 24] 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 [60, 104, 110, 75, 19]. Extending the scope of a web crawl has implications on the coverage of search engines and in web preservation [58, 77].

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 [16], and a combination of indexes and HTTP extensions [138]. WebDAV [41] provides some update semantics through

HTTP extensions, but has yet to find wide-spread adoption. The RSS syndication formats [112] 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 [45, 149], 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 [149]. Sites with OAI-PMH servers can register with MSN’s “AcademicLive” search service for enhanced content harvesting. Google, however, has recently announced its intention to drop support for the protocol [92].

The top three search engines, Google, Yahoo and MSN, recently agreed to support the *Sitemap protocol* which provides a very specific means to provide search engine crawlers with a list of crawlable resources [126]. These search engines also give webmasters a means to “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. For example, the Old Dominion University site (<http://www.odu.edu/>) has a link Although many optimization techniques have been proposed [24, 57], the task is still processor- and time-intensive.

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 [128]. 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 9 on the following page presents a transcript from an actual response-request sequence.

Obviously, this is markedly less efficient than an equivalent request for a list of pages that *have* been modified since that date. If a web site 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 30 on the next page 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.

TABLE 9: HTTP Request-Response Example

Request:	GET /index.html HTTP/1.1
	Host: www.modoi.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"

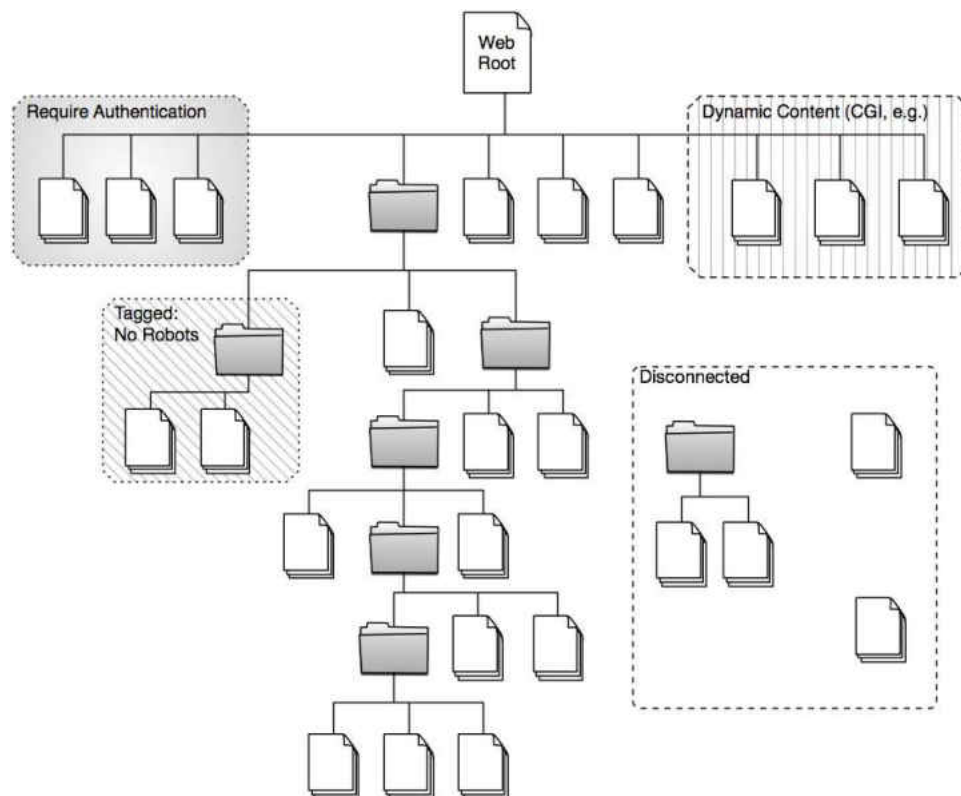


FIG. 30: Crawler's view of a website. The crawler does not "see" any of the content within the dashed box areas. The robots.txt file tells the crawler not to visit certain pages, even though they are accessible. Other pages require user authentication, and still others are dynamically generated (CGI, 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.

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 [30]. DASL’s goal was a lightweight 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 [13]. 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 [53]. 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 [16, 109] (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 web sites [128]. This behavior implies that crawlers expect the ratio of updates to be low relative to the total number of conditional requests issued.

3 THE SITEMAP PROTOCOL

To date, much of the research in the web community has focused on efficiently estimating updates and additions of remote, uncooperative web servers. Now, there is interest in shifting some (if not most) of the responsibility for resource discovery to the web servers themselves, via the *Sitemap*

```

<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. 31: Website URLs in a Sitemap file

protocol. Technically a file format rather than a true protocol, it was originally developed and promoted by Google [107, 45] 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 [126] 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 31 gives a brief example of a the urlset section of a Sitemap file. Appendix E on page 183 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. 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-built file; 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 always 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

Enumerating all of a site's resources is a challenge for both crawler and webmaster. The HTTP protocol does not provide a method for asking a site to "give me everything you've got." Page links, the common method for resource discovery, may not point to all of a site's resources. Sitemaps, created locally by the webmaster, are subject to the race condition between site changes and Sitemap file updates. In short, the counting problem is not solved by these approaches, it is only ameliorated.

CHAPTER VI

EVALUATION OF RESOURCE ENUMERATION METHODS

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 bB.html*, which does not exist. 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 and evaluate solutions for developing a canonical list of all URIs at a given site, the author used a copy of the ODU Computer Science Department (CS) website. A single snapshot of the CS website was provided for this experiment which was recovered from a backup tape by the website's system administrators. Several files have zero byte size, such as "maly3.jpg" and "New Text Document.txt". Errors like these in backup files are not uncommon. The snapshot has a datestamp of 06 June 2006. 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 particular file also fits into the "cruft" category (see Section 1.5 on page 75), 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.

1.2 The Website Structure

A graphic layout of the website is shown in Figure 32 on page 67. The website itself is neither very wide (only 16 directories at root level), nor very deep (5 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

TABLE 10: Composition of the CS Website according to the snapshot file tree. 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

be seen in the following sections, 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 10. As shown in Figure 32 on page 67, 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 on page 123 for more a 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 11 on the next page. 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 [71, 78, 23]. The department website 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.

TABLE 11: 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 filesystem. 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

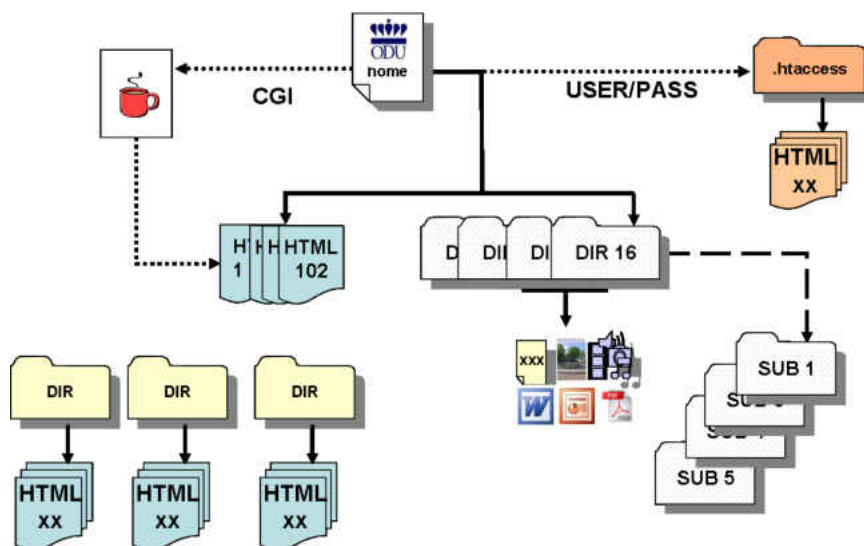


FIG. 32: Graphical View of the CS Department Website. The depth of the site is 5, and not all directories are linked even though they may be accessible if the path is known. There are some PHP and Perl scripts used to generate some pages and certain elements on other pages, as well as Javascript (which 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.

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.

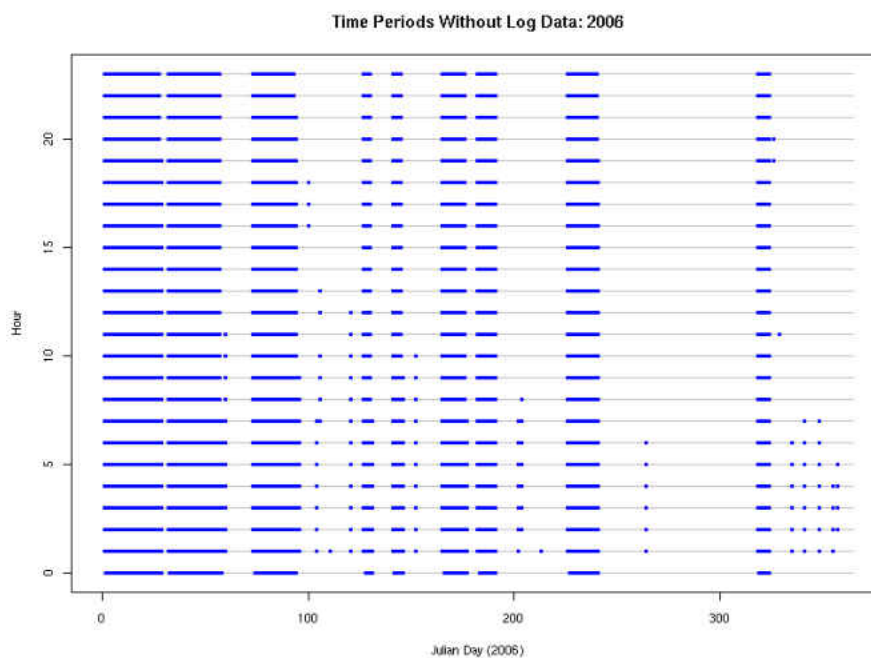


FIG. 33: Log coverage during the 2006 calendar year was very low, only 26%. In some cases, only a single hour of activity was logged on a calendar day.

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 33 and 34 on the next page for a graph of coverage during 2006 and 2007.

Despite these gaps, there are over 15 million *successful* requests in the 2-year period of the logs (unsuccessful requests were ignored, since they do not point to a valid resource). 3233 distinct resources (files, plus open directories, plus scripts by filename) 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 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. Despite this poor showing, 15,585,100 successful

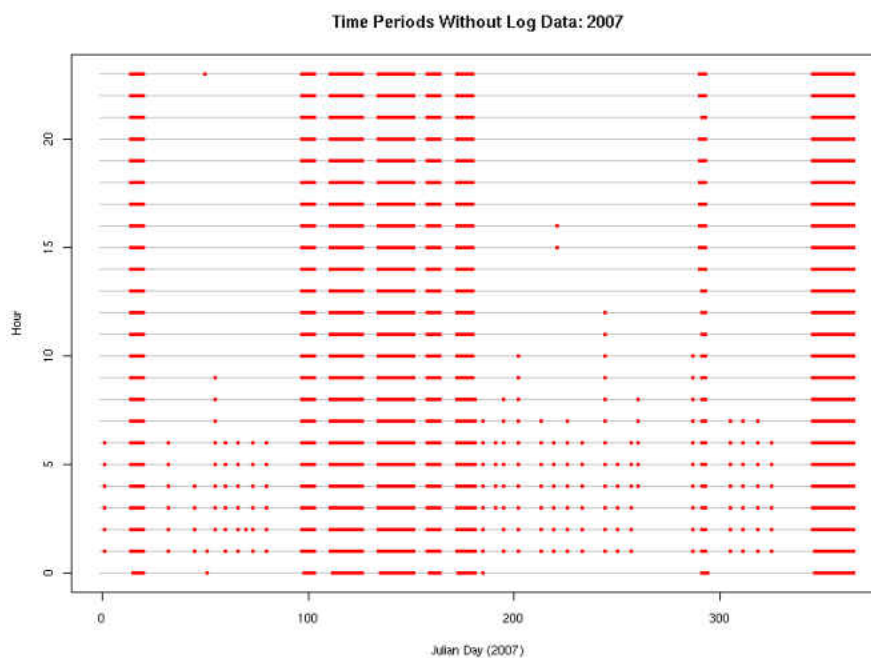


FIG. 34: Log coverage during the 2007 calendar year was better than for 2006, but still only 37%.

requests were extracted from the logs. Comparing the logged requests with the known file system resource list, 68% are logged. Figure 35 on the following page 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 36 on page 71 shows that the coverage rate reaches 98%, significantly better than the 68% shown in Figure 35 on the following page.

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 12 on the next page. 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 sinister “GET . . / . . / . . / . . / . . / . .

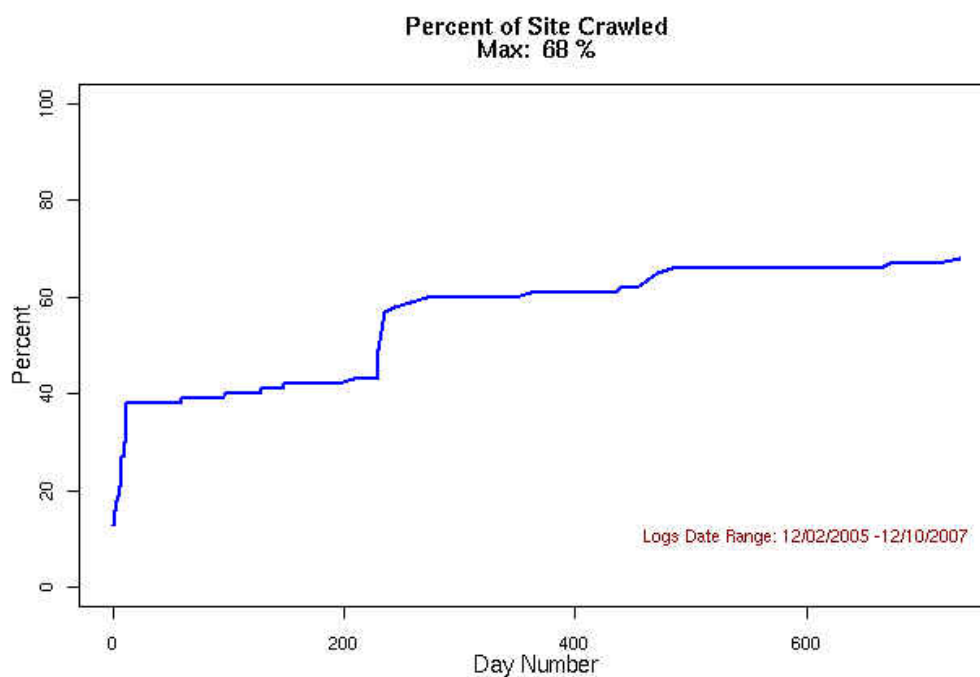


FIG. 35: This graph shows the proportion of *all accessible* website resources that were visited, either by a robot or by a user with a browser. Here “100%” includes items like backup files which, though accessible, are rarely if ever visited.

TABLE 12: Web server log fields 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 13 on page 73)
bytes	size of response in bytes

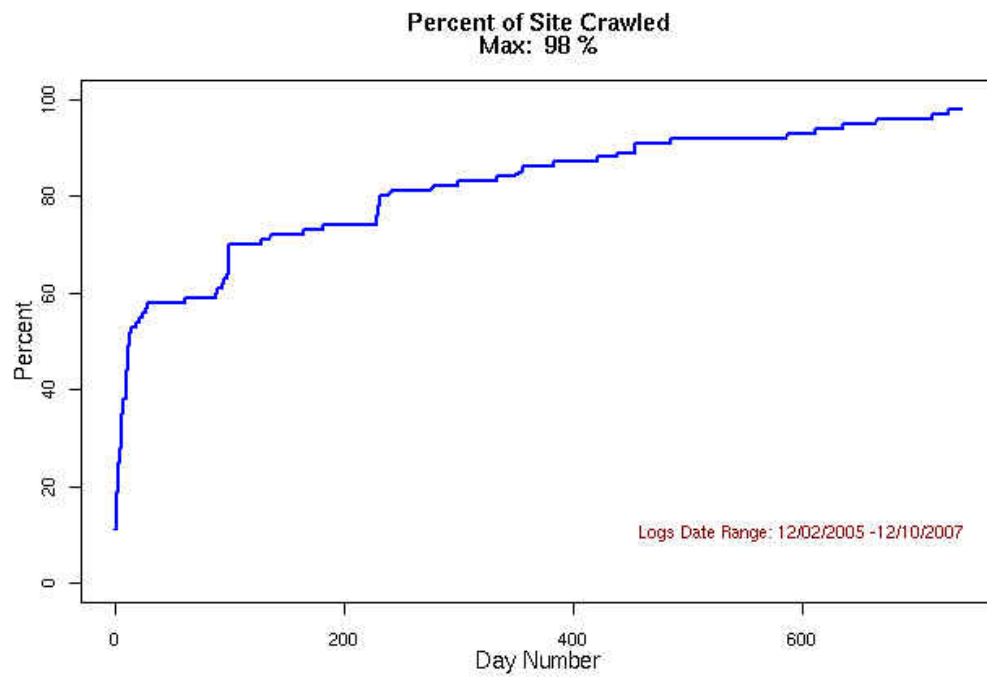


FIG. 36: This graph shows the percent of *expected* website resources that were visited, either by a robot or by a user with a browser. Here “100%” only includes “normal” files – those in publicly accessible directories that are not backup files.

`../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`) 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.

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 13 on the next page 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) and 403 (Access 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 14 on the following page shows the request distribution in the logs for each Method, grouped

TABLE 13: HTTP Response Codes 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 valid 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

TABLE 14: Request 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

by type of visitor. The main goal for the author was to separate normal browser usage of the site from everything else. In general, “everything else” is safely put in the “crawler” category since (a) browsers do not 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 15 on the next page and in Table 16 on the following page. 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 16 on the next page). 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 it is automatically sent to browsers in association with many other pages from the site.

TABLE 15: Request 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.

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

TABLE 16: Request distribution on the CS Website by type of visitor and application type of requested resource.

Application	Robots	Users	All
DOC	39,722	1,193	40,915
PDF	25,953	9,209	35,162
PPT	228	378	606
XLS	9,338	398	9,736
PS	14	4	18
MPG	115	61	176
MSO	2,681	27	2,708

TABLE 17: Resource duplication on a site 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 moved at some point. Consider, for example, the URLs in Table 17. The two URLs appear to be the same resource, having the same basic filename 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 17 fit the description of “DUST,” Different URLs with Similar Text [7]. 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) to dynamically-generated content (.shtml), which is primarily 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 35 on page 70 and Figure 36 on page 71.

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 on page 69, 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* log requests are not found in the snapshot. According to the website system administrators, Rewrite rules are applied to some requests, but the original 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 37 on the next page 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`) which

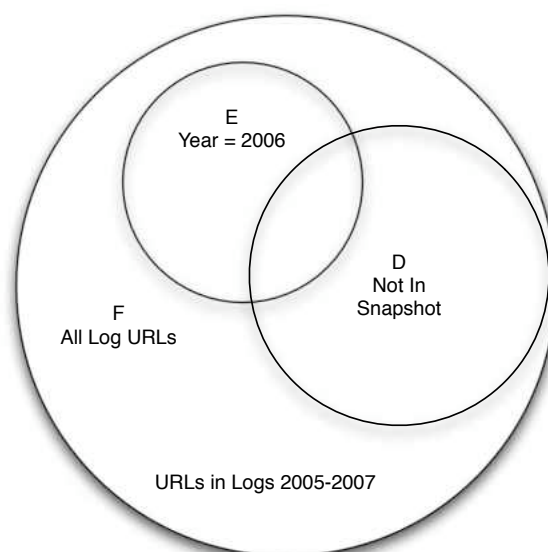


FIG. 37: F represents all unique resources from the 2006-2007 logs (which actually include part of 2005); $F = 3233$. E contains all the resources from the snapshot, dated 6 June 2006. $E = 2742$. Since the logs cover a wide period surrounding the snapshot, many resources appear in the logs that are 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; $E \cap D = 469$.

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 18.

The links listed in Table 18 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 19 on page 79.

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). 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.

A self-crawl produces directory hints, such as the “images/” and “files/” directories, which a

TABLE 18: Links on 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

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. Results of the self-crawl were poor: 538 distinct URLs which resolved to only 406 snapshot resources. Compare these numbers with Figure 38 on the next page which shows the relationship between the snapshot resources and those that were crawled at some point by users or robots.

2.3 File Tree VS Sitemaps Tools

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. Such tools also have options that allow specific dynamic URLs to be included, 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

TABLE 19: Fully-qualified internal links on the CS Website.

Request	Response
http://www.google.com/search?q=xxx&operation=1 &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 /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
<a href="http://www.cs.odu.edu/~home_g/grad_home/grad_info
/things_to_do_to_graduate.html">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

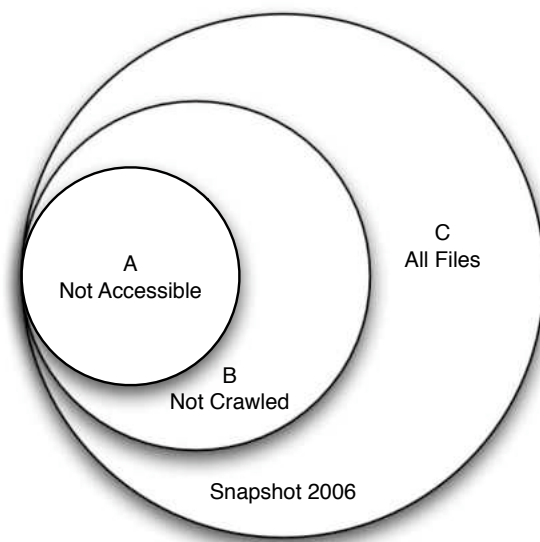
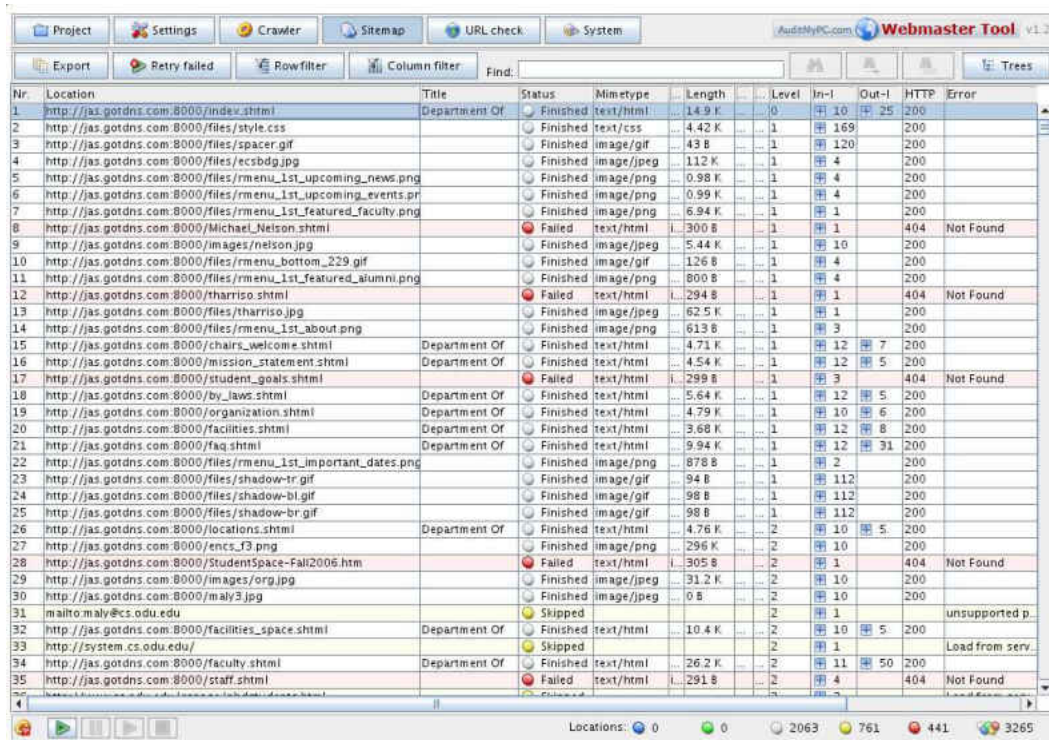


FIG. 38: The snapshot contains 2273 resources, $C = 2273$. Of these, $B = 722$ do not appear in any of the logs. The snapshot has some protected resources which are not accessible, $A = 577$, leaving $A - B = 145$ resources that were accessible but never crawled.



Nr.	Location	Title	Status	Mimetype	Length	Level	In-l	Out-l	HTTP	Error
1	http://jas.gotdns.com:8000/indav.shtml	Department Of	Finished	text/html	14.9 K	0	10	25	200	
2	http://jas.gotdns.com:8000/files/style.css		Finished	text/css	4.42 K	1	169		200	
3	http://jas.gotdns.com:8000/files/spacer.gif		Finished	image/gif	43 B	1	120		200	
4	http://jas.gotdns.com:8000/files/ecsbdg.jpg		Finished	image/jpeg	112 K	1	4		200	
5	http://jas.gotdns.com:8000/files/rmenu_lst_upcoming_news.png		Finished	image/png	0.98 K	1	4		200	
6	http://jas.gotdns.com:8000/files/rmenu_lst_upcoming_events.pr		Finished	image/png	0.99 K	1	4		200	
7	http://jas.gotdns.com:8000/files/rmenu_lst_featured_faculty.png		Finished	image/png	6.94 K	1	1		200	
8	http://jas.gotdns.com:8000/Michael_Nelson.shtml		Failed	text/html	300 B	1	1		404	Not Found
9	http://jas.gotdns.com:8000/images/nelson.jpg		Finished	image/jpeg	5.44 K	1	10		200	
10	http://jas.gotdns.com:8000/files/rmenu_bottom_229.gif		Finished	image/gif	126 B	1	4		200	
11	http://jas.gotdns.com:8000/files/rmenu_lst_featured_alumni.png		Finished	image/png	800 B	1	4		200	
12	http://jas.gotdns.com:8000/tharriso.shtml		Failed	text/html	294 B	1	1		404	Not Found
13	http://jas.gotdns.com:8000/files/tharriso.jpg		Finished	image/jpeg	62.5 K	1	1		200	
14	http://jas.gotdns.com:8000/files/rmenu_lst_about.png		Finished	image/png	613 B	1	3		200	
15	http://jas.gotdns.com:8000/chairs_welcome.shtml	Department Of	Finished	text/html	4.71 K	1	12	7	200	
16	http://jas.gotdns.com:8000/mission_statement.shtml	Department Of	Finished	text/html	4.54 K	1	12	5	200	
17	http://jas.gotdns.com:8000/student_goals.shtml		Failed	text/html	299 B	1	3		404	Not Found
18	http://jas.gotdns.com:8000/by_laws.shtml	Department Of	Finished	text/html	5.64 K	1	12	5	200	
19	http://jas.gotdns.com:8000/organization.shtml	Department Of	Finished	text/html	4.79 K	1	10	6	200	
20	http://jas.gotdns.com:8000/facilities.shtml	Department Of	Finished	text/html	3.68 K	1	12	8	200	
21	http://jas.gotdns.com:8000/faq.shtml	Department Of	Finished	text/html	9.94 K	1	12	31	200	
22	http://jas.gotdns.com:8000/files/rmenu_lst_important_dates.png		Finished	image/png	878 B	1	2		200	
23	http://jas.gotdns.com:8000/files/shadow-tr.gif		Finished	image/gif	94 B	1	112		200	
24	http://jas.gotdns.com:8000/files/shadow-bl.gif		Finished	image/gif	98 B	1	112		200	
25	http://jas.gotdns.com:8000/files/shadow-br.gif		Finished	image/gif	98 B	1	112		200	
26	http://jas.gotdns.com:8000/locations.shtml	Department Of	Finished	text/html	4.76 K	2	10	5	200	
27	http://jas.gotdns.com:8000/encs_f3.png		Finished	image/png	296 K	2	10		200	
28	http://jas.gotdns.com:8000/StudentSpace-Fall2006.htm		Failed	text/html	305 B	2	1		404	Not Found
29	http://jas.gotdns.com:8000/images/org.jpg		Finished	image/jpeg	31.2 K	2	10		200	
30	http://jas.gotdns.com:8000/maly3.jpg		Finished	image/jpeg	0 B	2	10		200	
31	mailto:maly@cs.odu.edu		Skipped			2	1			unsupported p.
32	http://jas.gotdns.com:8000/facilities_space.shtml	Department Of	Finished	text/html	10.4 K	2	10	5	200	
33	http://system.cs.odu.edu/		Skipped			2	1			Load from serv.
34	http://jas.gotdns.com:8000/faculty.shtml	Department Of	Finished	text/html	26.2 K	2	11	50	200	
35	http://jas.gotdns.com:8000/staff.shtml		Failed	text/html	291 B	2	4		404	Not Found

FIG. 39: Some sitemap tools provide sophisticated, GUI-based reports. This graphical website representation was produced by the Webmaster Tool on auditmypc.com

site-crawl, although sometimes a specific directory tree can be designated as long as it is directly web-accessible. For example, Webmaster Tool [4] 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 39) and creates reports in plain XML, sitemap-compliant XML, and HTML. Webmaster Tool cannot traverse the site 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 [150] 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 33 on page 68 and in Figure 34 on page 69 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

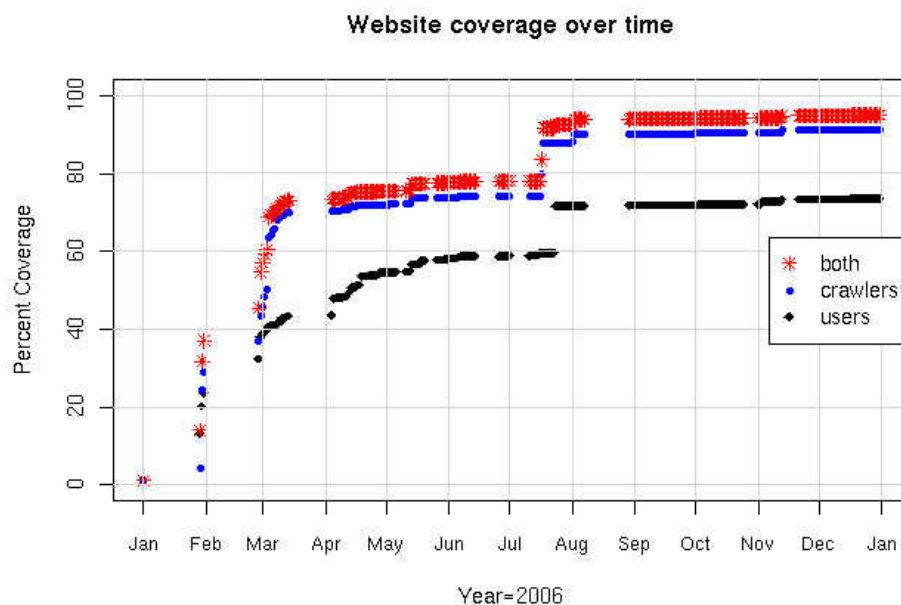


FIG. 40: Site Coverage from Web Logs

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 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 40 shows how the CS Department website logs come close to accessing all known snapshot resources.

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, requested by both crawlers and users as Table 20 on the following page shows.

TABLE 20: Request distribution on the CS Website by visitor and dynamic resource type. Type “Other” includes Java, Ruby On Rails, and Python.

Type	Robots	Users	All
PHP	8,525	37,261	45,786
PERL	3,313	7,834	11,147
CGI (other)	83	114	197
Other	264,949	287,709	552,658

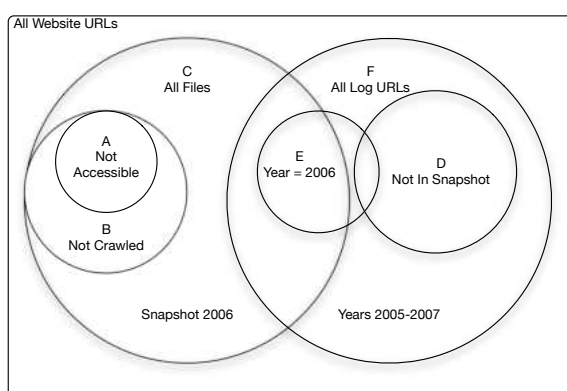


FIG. 41: Website Coverage from Integrated Counting Techniques. It is not possible to guarantee that all website resources will be counted, but more are likely to be found using this method than any single approach.

3 SUMMARY OF EXPERIMENT RESULTS

Self-crawls, file-system traversal, and log harvesting produce different lists of website resources. In Figure 41, a comparison of the results from these different enumeration methods shows the advantages of integrating all three approaches into a single solution. The universe of all website resources may be more than the sum of that the counting utilities produce.

4 RESOURCE ENUMERATION & 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 a necessary delay that occurs between *resource update* and *sitemap update*: a race condition 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. There is no way to overcome this problem. Taking a site off-line does not solve it, because an earlier sitemap request could still be used as the basis for a later harvest. To some degree, there will always be a degree of uncertainty regarding the content of a site and the accuracy of its sitemap.

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 filenames, 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, will produce the most complete enumeration of resources.

This is not a one-time event, however. The process should be repeated periodically to capture new resources. 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. The Counting Problem does not have an easy solution.

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. In Figure 42, the digital object is the body of the HTTP response (JPEG content, for example). The representation information comes from the MIME type and other HTTP response headers, and the knowledge base (rendering tool) for presenting the object is built into the browser. The result is what OAIS terms an *informatio object*. Most browsers can appropriately display or describe today’s resources at the point of request. 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. The relationship between OAIS and the browsing experience is shown in Figure 42.

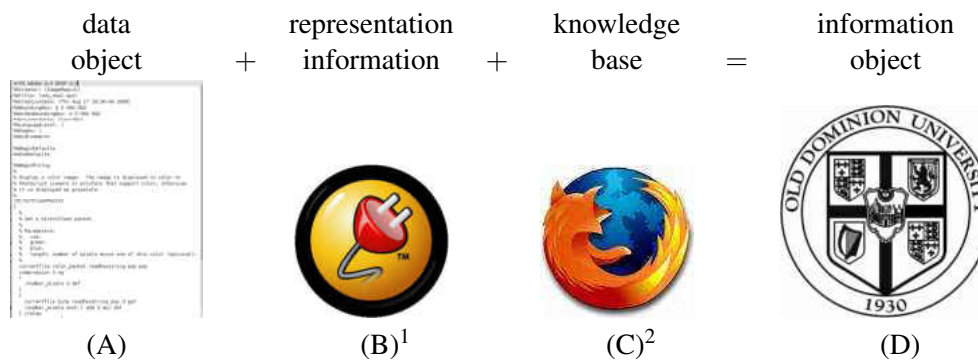


FIG. 42: Resource representation in the OAIS Model and from a browser’s perspective.

¹<http://jimmac.musichall.cz/images/logo/connector.png> ²<http://mozilla.org>

The Representation Problem is the need to have sufficient information to correctly understand the resource at some future point in time. Can a set of functions be defined that will preserve web site 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 a format that is understandable by the future system. The goal is to define a *Preservation* function (P) that archives the resource together with essential metadata; a *Restoration* function, R_e , which can reproduce the original via emulation, or which restores it via the migration of the content to the newer format (R_m):

$$\begin{array}{l} P(W) \xrightarrow{\text{archive}} \boxed{W} \\ R_e(\boxed{W}) \xrightarrow{\text{emulation}} \boxed{W} \\ R_m(\boxed{W}) \xrightarrow{\text{migration}} W_{\Delta} \end{array}$$

In brief, the Representation Problem is the problem of collecting sufficient descriptive information for each resource on a website so that it can be properly represented in the future.

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 interventions 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 be 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.

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 [121, 134, 148]. Google's approach was to weight links on web pages to produce a hierarchy 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 43 on the next page, 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 doesn't 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 [80]. 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 [103, 90]. On the other hand, sometimes the intent of text is to communicate a picture, as in Figure 44 on page 88. How can this representation be distinguished from the spam-like content of Figure 43 on the next page? How 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.

Search engines also alter content representation when they transform the site resource in the cached copy they keep. Consider Figure 45, 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

```

<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><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
</BODY></HTML>

```

(A) Crawler-Viewed Content



(B) Browser-Viewed Content

FIG. 43: 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.

(A) Coffee Shop Zombies¹(B) Turmoil²

FIG. 44: ASCII Art 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. ¹http://www.penguinpetes.com/images/BBS_art/thumbs/Coffeeshop_Zombies.jpg ²http://www.penguinpetes.com/images/BBS_art/ASCII/Turmoil.jpg

the “recognition” factor for the client – this cached copy is less likely to be useful. Representation issues impact search engines as well as preservationists.

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 15 on page 33).

The MIME specification (Multipurpose Internet Mail Extensions) and MIME types 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 [37]. The MIME specification has enjoyed a nearly universal implementation, but it differentiates file content types on only a very simple level, and

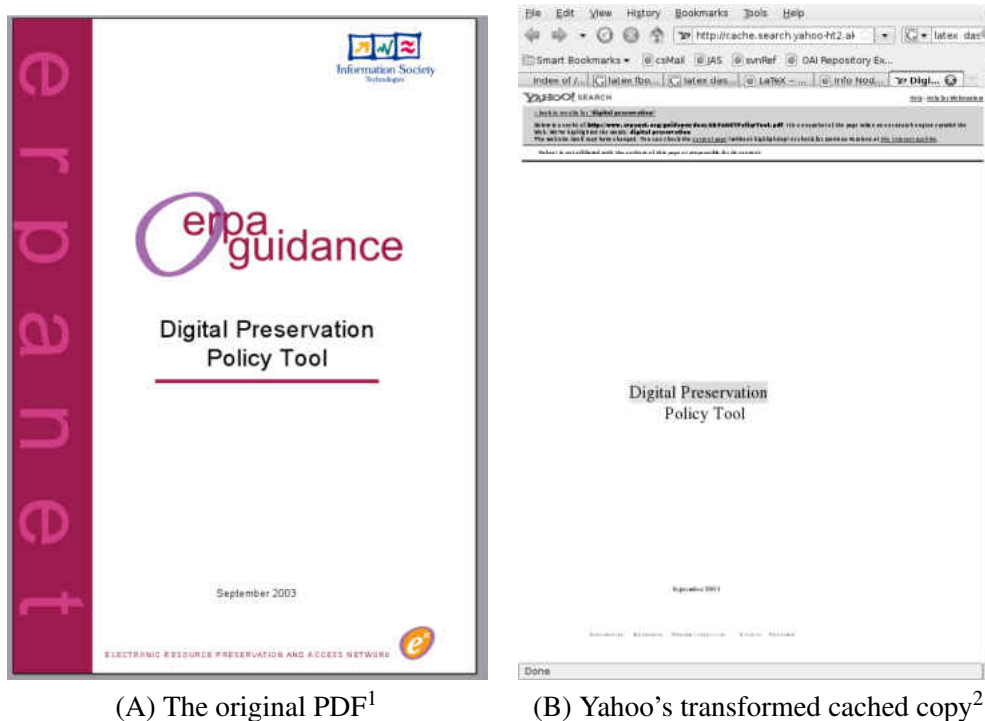


FIG. 45: Search engines sometimes transform resources that will be stored in cache. In the process, images and other information may be lost. ¹<http://www.erpanet.org/guidance/docs/ERPANETPolicyTool.pdf> ²<http://cache.search.yahoo-ht2.akadns.net/search/cache?ei=UTF-8&p=digital+preservation&y=Search&fr=yfp-t-501&u=www.erpanet.org/guidance/docs/ERPANETPolicyTool.pdf&w=digital+preservation&d=Q1PzJpzfQw6A&icp=1&.intl=us>

one which is insufficient for archiving purposes. RFC 2046 defined 5 basic content types [38], and two composite types. The 7 categories are listed in Table 21, 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 voice messages is multipart 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.

MIME Type	Selected Subtypes
<i>Basic (Discrete) Types</i>	
(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
<i>Composite Types</i>	
(6) multipart	header-set, digest, mixed
(7) message	external-body, news, partial

TABLE 21: The MIME Content Type Categories

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 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 15 on page 33 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.

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 has 47 defined Headers which are grouped into 4 general categories: (1) Entity (2) General (3) Request and (4) Response. Table 22 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.

TABLE 22: HTTP Headers, 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	

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.

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” [122]. 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 doesn't fulfill repository "requirements." Another observation from the AIHT is that metadata markup, like ontologies, will never evolve into a universally-accepted approach [123]. 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,

$$DataObject + RepresentationInformation + KnowledgeBase = InformationObject$$

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 provide little metadata, and sometimes website resource content incorrectly influences 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

EVALUATION OF METADATA UTILITIES ON THE WEB SERVER

1 A REPRESENTATION PROBLEM EXPERIMENT

1.1 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 [6] notes, research on average website content is biased by the sampling method used, which is often either too random (sampling from search engines) or too focused (a single known site). An extensive survey of web content was published by Berkeley in 2003 [78]. 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 [6].

1.2 Metrics of Website Composition

With regard to website size, a 2004 report on the composition of various national domains [6] 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 [71] 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 [35].

A relatively recent (2004) examination of e-commerce sites at a large server farm [10] 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 [102, 23], 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.

1.3 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 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 [71] and [78].

Figure 46 on the following page 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 47 on the next page. 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 48 on page 96 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 23 on page 97 describes

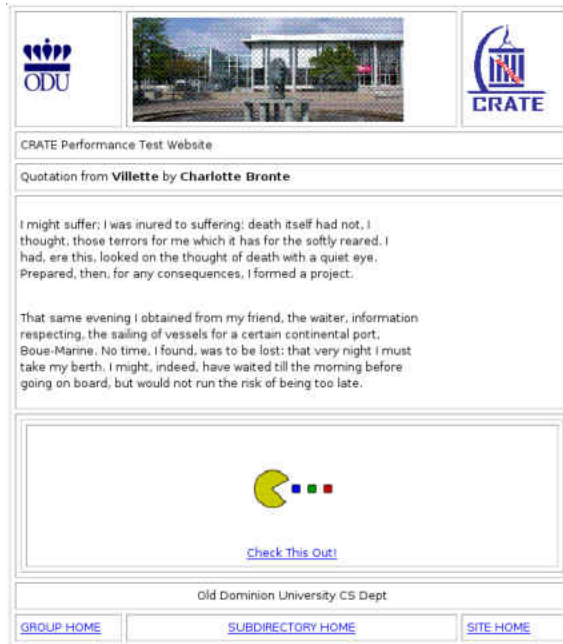


FIG. 46: An example page from the test website, showing a randomly-assigned resource (in this case, an embedded image).



FIG. 47: An example PDF from the test website. PDFs were generally 1-2 pages in length.

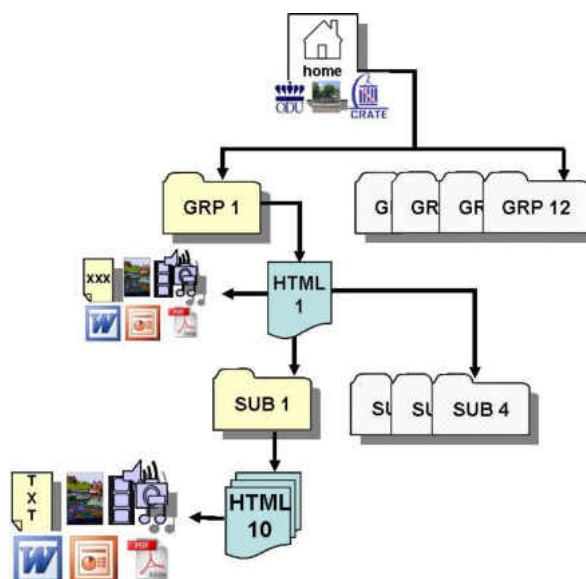


FIG. 48: Visual map of the test website. Resources of all types were distributed at all levels of the site.

the overall content of the site by file type and hierarchy. Table 24 on the following page shows the resource distribution by type and size for the test website.

1.4 Selecting Metadata Utilities

The target environment for this test is the small to mid-size 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 in mind, four elements were defined as the primary selection factors for the metadata utilities to be included in the test:

- (1) Price
- (2) Operating System
- (3) Invocation Method and
- (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

TABLE 23: Content and layout of 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
<i>Other^b</i>	$\leq 1^a$	$\leq 1^a$	49 (Total)
Total Files:			1084

TABLE 24: Distribution of resources in the test website by MIME type and size.

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

module used to enable metadata-utility integration (MODDOI) 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 D on page 178). 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.

1.5 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 [119]) 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 ([51, 14]) 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 ([64, 22]), 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. Exif, 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` and
- (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


```

<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. 49: Example of the META tag content extracted from one of the test web’s HTML pages using the author’s own “home-grown” utility, dcTag.

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 web-server 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 like the actual example shown in Figure 49. 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.

1.6 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 25 on the next page. Each of these not only meets the selection criteria but it also provides useful preservation-relevant metadata.

TABLE 25: Metadata 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 (<code>/usr/bin/exif</code>) Compiled version. Digital photo analysis	image/jpeg
WC	Word Count 0	Linux utility (<code>/usr/bin/wc</code>) 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 (<code>/usr/bin/file</code>) 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 (<code>/usr/bin/md5sum</code>) Digital file signature based on MD5 hash	*/*
SHA	SHA-Hash 0	Linux utility (<code>/usr/bin/shasum</code>) Digital file signature based on SHA hash	*/*
SHA-1	SHA1-Hash 0	Linux utility (<code>/usr/bin/sha1sum</code>) 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 26: Components of a plugin specification in the Apache configuration file.

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

1.7 Configuring The Web Server

As the most-installed web server brand in the world [101], 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 [99], 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 50 on the following page, and the components of a “plugin” specification are shown in Figure 26. In other respects, the Apache configuration options were left in the default state. No rewrite rules, aliases, or customizations were configured, except for the installation and initialization of MODOAI.

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.

1.8 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 live Internet traffic website, pictured in Figure 51 on page 103. 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

```

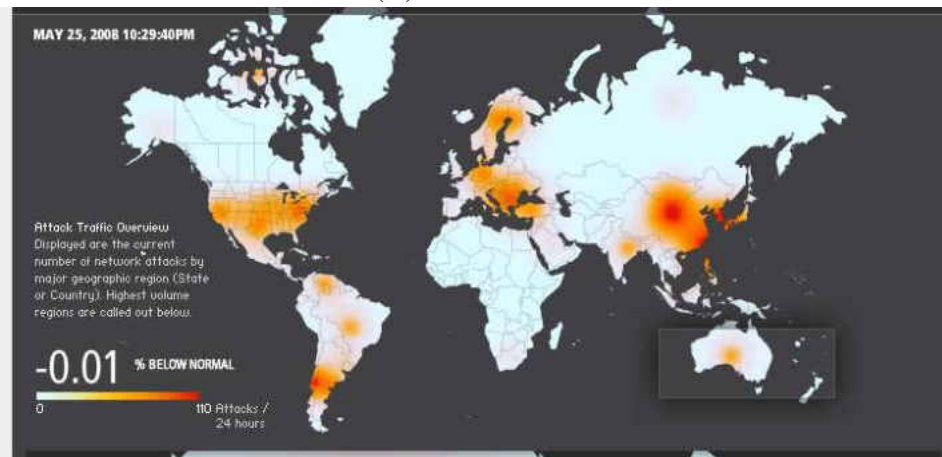
Alias /modoai "/var/www/"
<Location /modoai>
SetHandler          modoai-handler
modoai_sitemap      /var/www/sitemap.xml
modoai_admin        smith
modoai_email        admin@foo.edu
modoai_gateway_email mail@foo.edu
modoai_oai_active   ON
modoai_max_response_size 10000
modoai_max_response_items 10000
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_SignatureFile_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. 50: 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 config file.



(A) Active Traffic



(B) Active Attacks

FIG. 51: Akamai maintains a live view of web traffic, including overall traffic (A) and areas experiencing attacks (B). See <http://www.akamai.com/visualize>

“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

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 [10, 23] were done with an eye toward improving delivery services via CDNs and Application Servers. An unexpected result of [10] was the discovery that improper cookie use created unnecessary work for the servers. Other website traffic studies [35, 26] 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, show a Pareto 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 [23, 10, 6]. 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 [10] 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.”

2 PERFORMING THE EXPERIMENTS

2.1 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.

2.2 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 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 [10] 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.

2.3 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 [68], 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 on page 9 for a discussion of the model). This command would produce a list of repository items and the MARC metadata for each of them:

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

¹<http://jakarta.apache.org/jmeter/>

<pre> <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> </pre>	<pre> <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[e1f66cd707c2df36dafa8557d82536a1]]> </crateplugin:content> </crateplugin> </pre>
--	---

FIG. 52: A sample section of CRATE XML generated by modoai.

OAI-PMH also offers the ability to *harvest the resources themselves* and not just the metadata [136]. 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 [8]. 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 52. To mimic non-OAI-PMH harvesters, the “wget” utility was used.

3 A QUANTITATIVE COMPARISON OF UTILITY PERFORMANCE

3.1 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 “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.

TABLE 27: 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

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 27. 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

TABLE 28: Web server performance for full 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 25 on page 100.

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

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.

3.2 Performance Data

The test results in Table 28 show that even a modest web server can provide CRATE-type output without significantly impacting responsiveness. Table 28 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 (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 28, 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.

4 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 safe to generate metadata directly from the web server?
- (2) Is it safe 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 to the harvester and the practicality of requesting and receiving such large complex-object responses.

The data indicates that it is safe 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 safe 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 D on page 178). 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.

5 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, the best solution is to have as much metadata from as many sources as possible.

6 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 utilities 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 safe to both generate such metadata and to ask the web server for it, within certain parameters. Anything that can run automatically is likely to be compatible, although utility speed and CPU demands may make a utility infeasible. 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 IX

CRATE: A MODEL FOR SELF-DESCRIBING WEB RESOURCES

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

- (1) The Counting Problem (Resource Enumeration)
- (2) The Representation Problem (Resource Description)

The first problem, resource enumeration, depends in large measure on webmaster cooperation and expertise. This is due both to the nature of website structure as much as to the inherent limitations of HTTP. Where a simple SQL-like query may suffice to bring the full scope of a database's resources to view, HTTP does not have an equivalent form. There is no *GET */** in the HTTP protocol; it is designed to request a single resource. Full site resource enumeration has to be achieved by other means, and a metadata-rich description of any one website resource must be obtained by HTTP-workarounds.

The CRATE model addresses the enumeration problem by specifying a *Sitemap* document as the source of valid resources for a website. Any implementation of the CRATE model depends on the existence of a Sitemap to effect a full crawl of a website. To meet the enumeration requirements of the CRATE model,

- (1) A Sitemap (*Sitemap.xml*) must exist
- (2) The Sitemap must contain a canonical form for each valid, accessible site resource
- (3) The Sitemap must be available either directly (via an HTTP GET *Sitemap.xml* request) or indirectly (such as via an OAI-PMH ListIdentifiers request).

The research presented by this author addresses the representation problem by defining a web-preservation object 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 53 on the following page. 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* for automated, metadata-rich preservation-oriented harvesting of web resources.

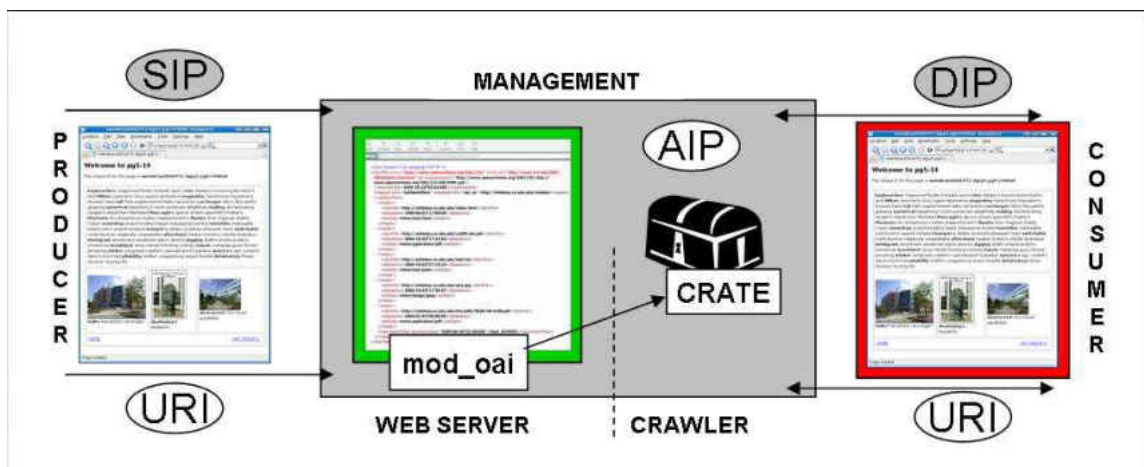


FIG. 53: CRATE Process in OAIS Context

- **automated:** CRATE uses the features and capabilities of the web server to perform the work automatically with each OAI-PMH response via the MODDOI module
- **metadata-rich:** Output from metadata extraction tools incorporated as plugins to MODDOI 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.

1 CRATE: A DATA-CENTRIC PRESERVATION MODEL

Stewart Brand's comment that we need data to be "born archival", not just digital [15] 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 [129], 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 [97, 136, 127]. 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 on page 12. 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 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. Similarly, the archiving repository would not need to worry about non-librarian webmasters misusing METS headers or MPEG-21 descriptors. Instead, 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 [123]. 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 a CRATE is simultaneously the DIP, AIP, and SIP. A conceptual mapping of CRATE onto the OAIS model is shown in Figure 54 on the following page. 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.

2 COMPLEX OBJECTS AS ARCHIVAL INFORMATION PACKAGES (AIPS)

Looking at a web site from the viewpoint of the OAIS [20, 79] model, a web site 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

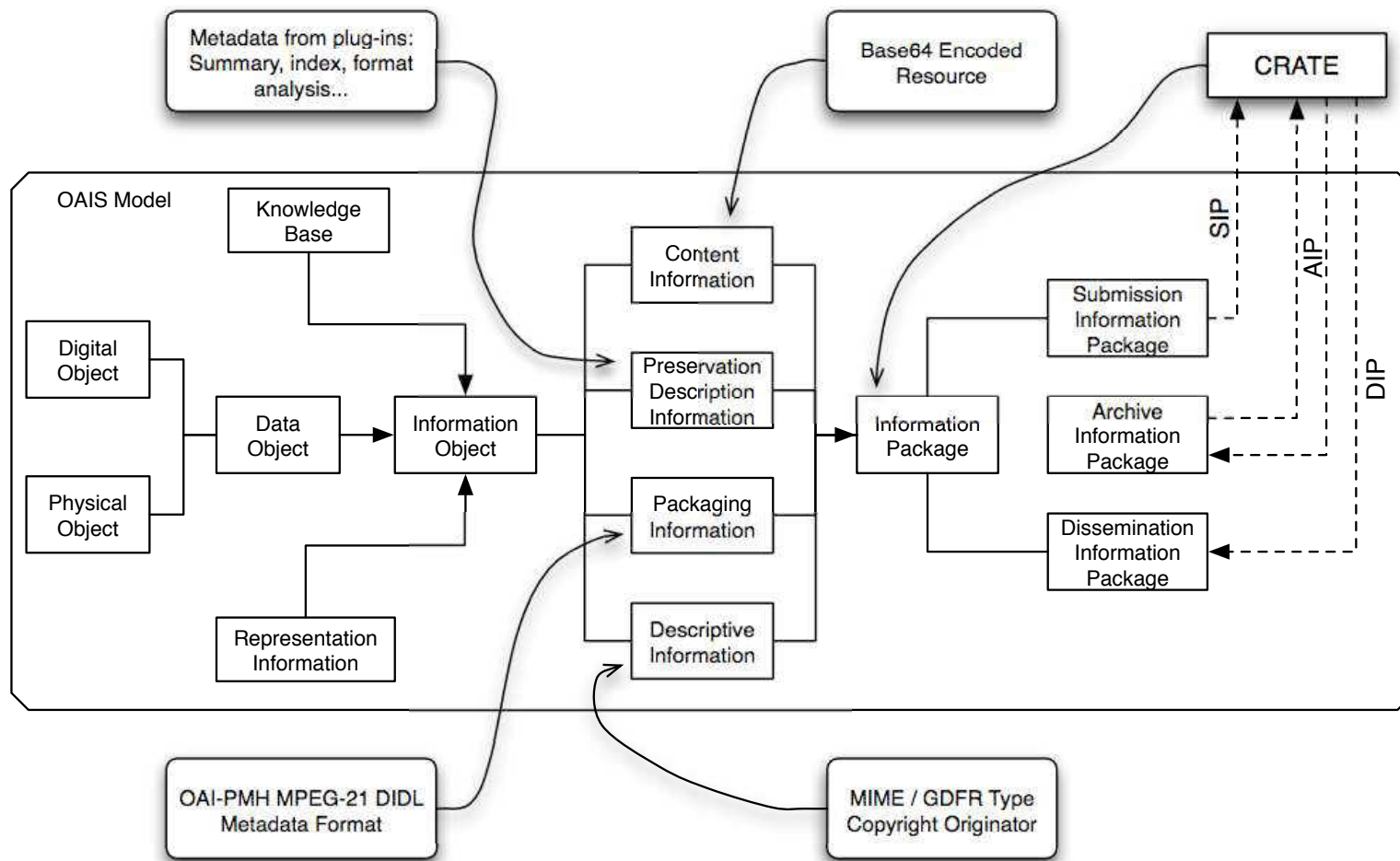


FIG. 54: CRATE in the OAIS Model

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 55 on the next page 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 [81], but the method for harvesting and packaging the object is different.

In Figure 55 on the following page 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. Response content transcribed from an actual URL is in Appendix 2 on page 160. 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 evidence that could be analyzed in the future, making the object once again accessible. Details of an example GetRecord response with DIDL metadata format content is in Appendix 2 on page 160.

Various models could be tested for compatibility with this concept: older approaches like Harvest [13], and more recent utilities such as RSS [108] and Sitemaps [45]. We would like to build on the efforts of previous researchers [97, 67, 136, 9], especially OAI-PMH, while also integrating the OAI-PMH concept of a DIP (Dissemination Information Package) [20]. OAI-PMH is a very powerful reification tool. Consider Figure 55 on the next page, 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



(A) Resource seen in a Firefox browser window

```

File Edit View Help
<?xml version='1.0' encoding='UTF-8'?>
<OAI-PHI 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-PHI.xsd'>
  <responseDate>2007-01-17T13:01:31Z</responseDate>
  <request verb='GetRecord' identifier='http://beatitude.cs.edu.edu:9999/oduLion.jpg'
  metadataPrefix='oai_crate' http://beatitude.cs.edu.edu:9999/crate/</request>
  <record>
  <header>
  <identifier>http://beatitude.cs.edu.edu:9999/oduLion.jpg</identifier>
  <datestamp>2007-01-17T04:09:07Z</datestamp>
  <setSpec:ieee:image/jpeg</setSpec>
  </header>
  <metadata>
  <did:BIID xmlns:did='urn:epg:peg21:2002:02-BIID-NS' xmlns:xsi='http://www.w3.org/2001/XMLSchema-
Instance' xsi:schemaLocation='urn:epg:peg21:2002:02-BIID-NS http://purl.lanl.gov/STR-PL/schemas/2004-11
/DIID.xsd'>
    <did:Itwp>
    <did:Descriptor>
    <did:Statement mimeType='application/xml; charset=utf-8'>
    <did:Identifier xmlns:did='urn:epg:peg21:2002:01-DID-NS' xmlns:xsi='http://www.w3.org/2001/XMLSchema-
Instance' xsi:schemaLocation='urn:epg:peg21:2002:01-DID-NS http://purl.lanl.gov/STR-PL/schemas/2003-09
/DID.xsd' http://beatitude.cs.edu.edu:9999/oduLion.jpg</did:Identifier>
    </did:Statement>
    </did:Descriptor>
    <did:Statement mimeType='application/xml; charset=utf-8'>
    <oai_dc:dc xmlns:oai_dc='http://www.openarchives.org/OAI/2.0/oai_dc/' xmlns:dc='http://purl.org
/dc/elements/1.1/' xmlns:xsi='http://www.w3.org/2001/XMLSchema-
Instance' xsi:schemaLocation='http://www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI/2.0
/oai_dc.xsd'>
      <dc:date>2007-01-17T04:09:07Z</dc:date>
      <dc:format>image/jpeg</dc:format>
      <dc:identifier>http://beatitude.cs.edu.edu:9999/oduLion.jpg</dc:identifier>
    </oai_dc:dc>
    </did:Statement>
    </did:Descriptor>
    <did:Statement mimeType='application/xml; charset=utf-8'>
    <plugin:file>
    <![CDATA[JPEG image data. JFIF standard 1.00. resolution (DPI), *LEAD Technologies Inc. V1.0].
    ]]>
    </plugin:file>
    </did:Statement>
    </did:Descriptor>
    <did:Statement mimeType='application/xml; charset=utf-8'>
    <plugin:info>
    <![CDATA[None (Rel. 1.1. 2006-06-05)
    Date: 2007-01-17 07:59:27 EST
    RepresentationInformation: /home/crate/apache/htdocs/oduLion.jpg
    ReportingModule: JPEG-hul Rel: 1.2 (2005-09-22)
    LastModified: 2007-01-16 23:00:07 EST
    Size: 27750
    Format: JPEG
    Version: 1.00
    Status: Well-Formed and valid
    SignatureMatches:
    JPEG-hul
    MIMEtype: image/jpeg
    Profile: JFIF
    ]]>
    </plugin:info>
    </did:Statement>
  </did:BIID>
  </metadata>
</record>
</OAI-PHI>

```

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

FIG. 55: 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.

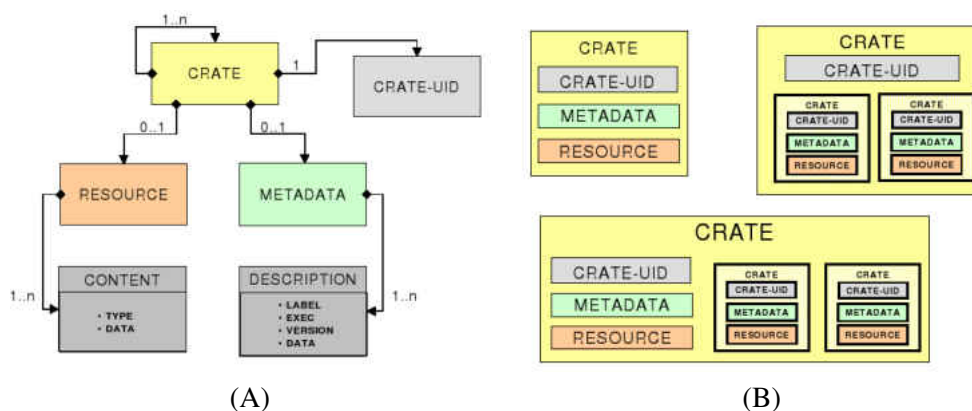


FIG. 56: The CRATE model and example CRATE configurations

the resource, as depicted in Figure 55 on the preceding page.

Of course, these are browser-views of the event, but they illustrate our concept of providing more comprehensive information to a crawler. In response to a search engine robot's request for the URL, the web server will return an OAI-PMH GetRecord request for that object. The result is a package that is a complete Dissemination Information Package (DIP) encoding for that URL.

3 BUILDING THE CRATE

An important characteristic shared by the models discussed in Chapter II on page 9 is that they are mostly human-readable, plain ASCII. 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 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. A conceptual view of a CRATE is shown in 56.

Only 3 elements are defined for a CRATE:

- Identifier
- Metadata
- Resource

Figure 56 expresses the CRATE complex object as a UML diagram. 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 [123]. 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. Resource disambiguation between repositories that have crawled the same sites can be done using the metadata elements of the CRATE.

Metadata is the heart of CRATE. Our goal is to automate the resource-description process; to have resources describe themselves in type-appropriate and sufficient detail; and to lower the barrier to preservation by simplifying participation requirements while maximizing resource information. Just as content types and versions vary from web site to web site, 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.

The Resource component of a CRATE can have one or many expressions. Each expression contains a `TYPE` element which describes the kind of content (Base64 encoding, text/html, etc.), and a `CONTENT` element with 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 and as Base64, for example, but it is otherwise idempotent. De-transformation of the content should produce a duplicate of the original resource. 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.

Example CRATE configurations are shown in Figure 56 on the previous page. 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 web site; or to keep an HTML resource together with its embedded multimedia content.

4 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.

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.

5 IMPLEMENTING THE CRATE MODEL

CRATE addresses both the Counting Problem and the Representation Problem. 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) List of unique site resources
- (2) Access to a site resource listing
- (3) Metadata utilities appropriate to site file types
- (4) Base64-encoding utility

The first two components are designed to address the Counting Problem; the second two address the Representation Problem.

5.1 Implementing the Counting Problem Solution

CRATE requires a site to have a list of all its resources, uniquely itemized. “All” in this case refers to those resources that are part of the website. Backup files, for example, may or may not be included; the website controls its own definition of “all.” Each resource must be unique in the list, i.e., no duplicate URIs. This allows the response to contain the required Unique ID.

The list must be *accessible*, but the means of access can depend on the site. It could be *direct*, where the file itself is exposed on the website as a resource; or *indirect* where a particular GET request produces some alternative format of the listing as a response (an OAI-PMH ListIdentifiers request, for example). The purpose of accessibility to the listing is that it clearly defines the resource scope of the site.

5.2 Implementing the Representation Problem Solution

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.

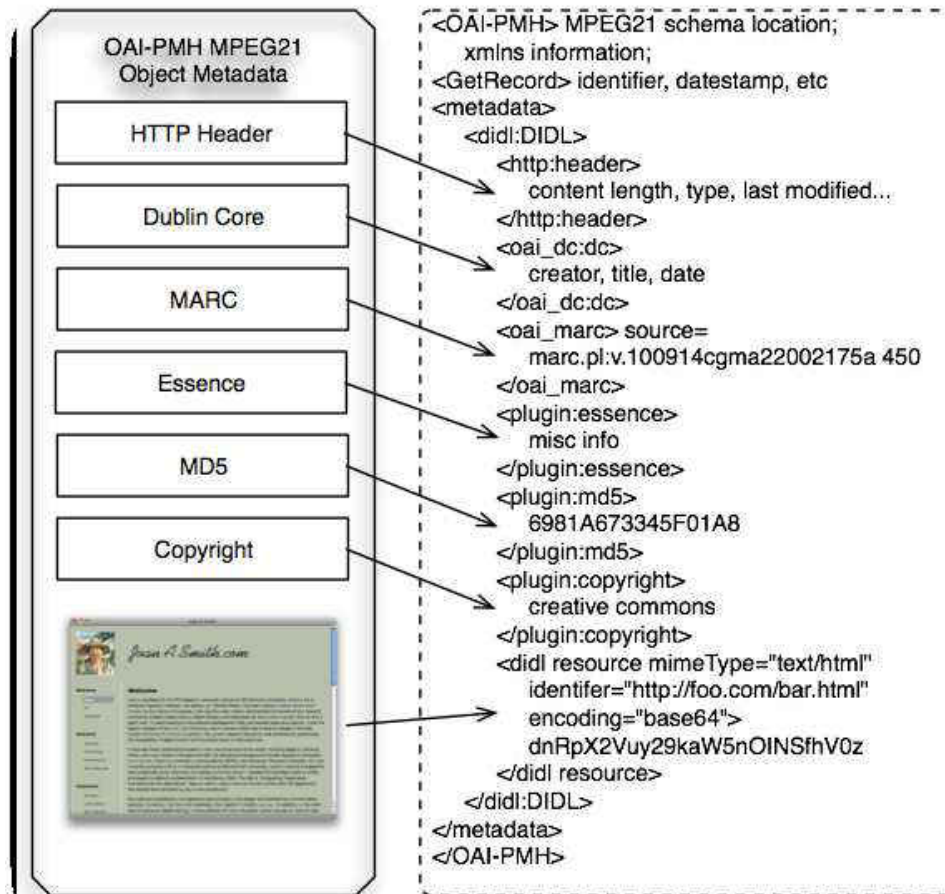


FIG. 57: Conceptual View of CRATE Object and XML Output in the MPEG-21 DID format.

5.3 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 define a particular string as an indicator that the client is asking for the CRATE form of the resource: *GET /resourceName?type=crate*, or whatever other method suits the Repository, its service provider, and harvesters. 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.

6 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 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 X

MODOAI: AN IMPLEMENTATION OF THE CRATE REFERENCE MODEL

As part of this dissertation’s research, a full implementation of the CRATE Reference Model was undertaken. It 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. 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 by the author to enable CRATE implementation and facilitate ongoing development of the software.

This chapter also describes the architecture of MODOAI itself, how it differs from the original version, and the processing loop as it occurs within the Apache request life cycle. The author then describes a series of experiments implementing MODOAI with a variety of metadata utilities installed on a web server, and evaluates the performance and feasibility of this model under such an implementation.

1 BACKGROUND: THE APACHE WEB SERVER

The Apache web server has been the dominant web server platform for over 10 years [101]. Apache’s success can be attributed to many things, including solid security, reliability, and a modern plugin-based architecture which easily allows the server to be extended to provide new functionality. As an Open Source product, Apache has benefited from the scrutiny of innumerable software engineers, making it an extremely secure and reliable product.

1.1 Web Server Modules

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 modules architecture allows developers to hook directly into the Apache request processing loop, and 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.

The Apache server runs as a server process, listening for new connections from web clients. When a new request arrives, Apache spawns a new process or thread, depending on the underlying operating systems capabilities. That process or thread is responsible for generating a response,

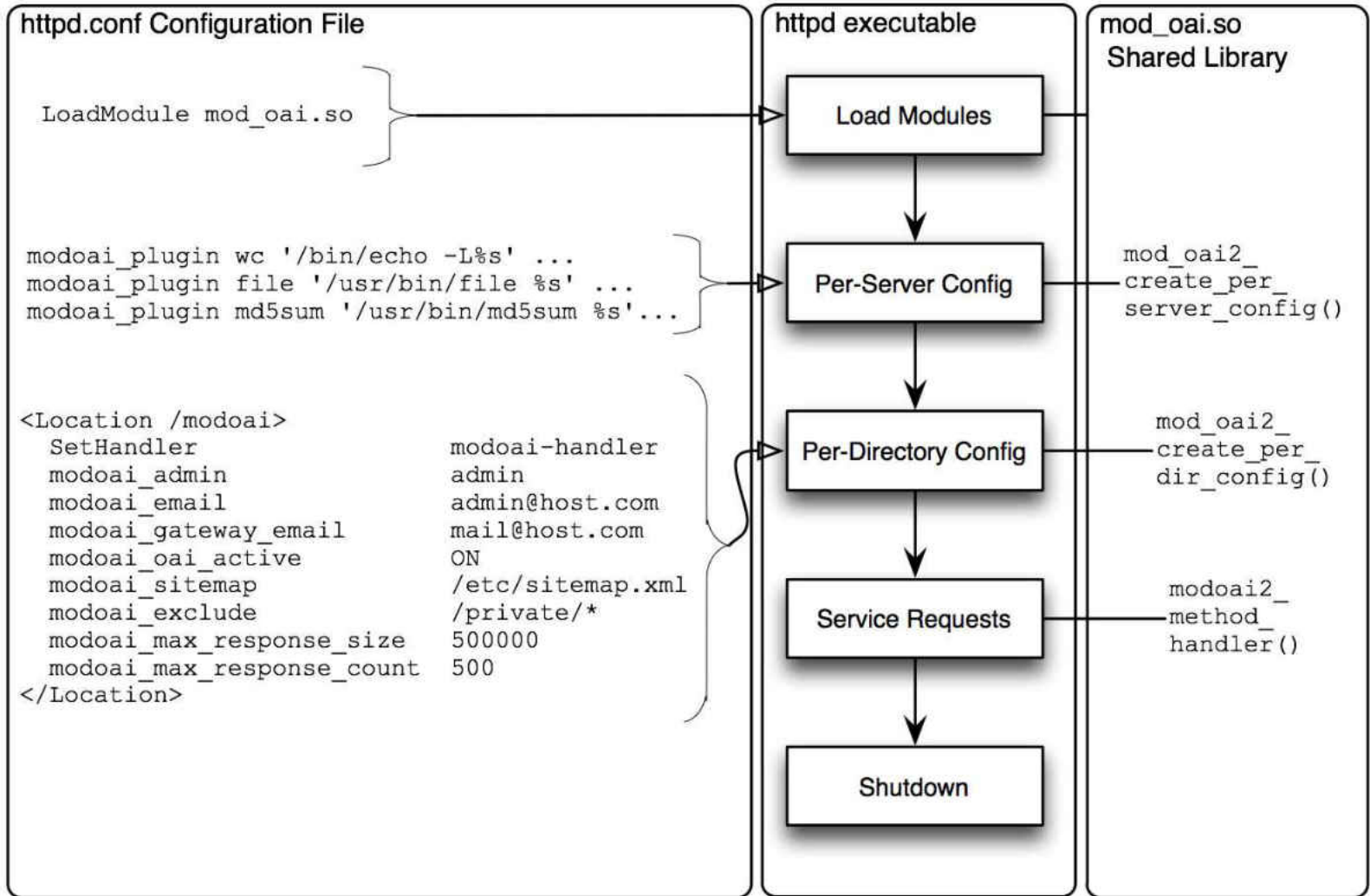


FIG. 58: The Apache Web Server Life Cycle.

which is often a 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. 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. (*Cf.* Figure 59 on the next page).

Conceptually, these phases can be simplified into groups accomplishing five tasks:

- (1) What is being requested
- (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.

Depending on their purpose, different Apache modules 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 MODDOI module registers itself to participate in the content generation phase. Once registered, MODDOI will be called for all requested URIs which have been mapped to MODDOI in the Apache configuration file.

By convention, there is one canonical URI for the MODDOI handler, /moddoi.

1.2 Platform Independence

Apache modules make heavy use of the Apache Portable Runtime to achieve platform independence. APR is an independent Apache project started to isolate the Apache HTTP server from the differences in the underlying operating systems. The MODDOI module makes heavy use of the APR functions. A few of these key functions MODDOI 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 MODDOI, 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 MODDOI code uses apr_tables to store two important sets of data: the list of query parameters from the request URL, and

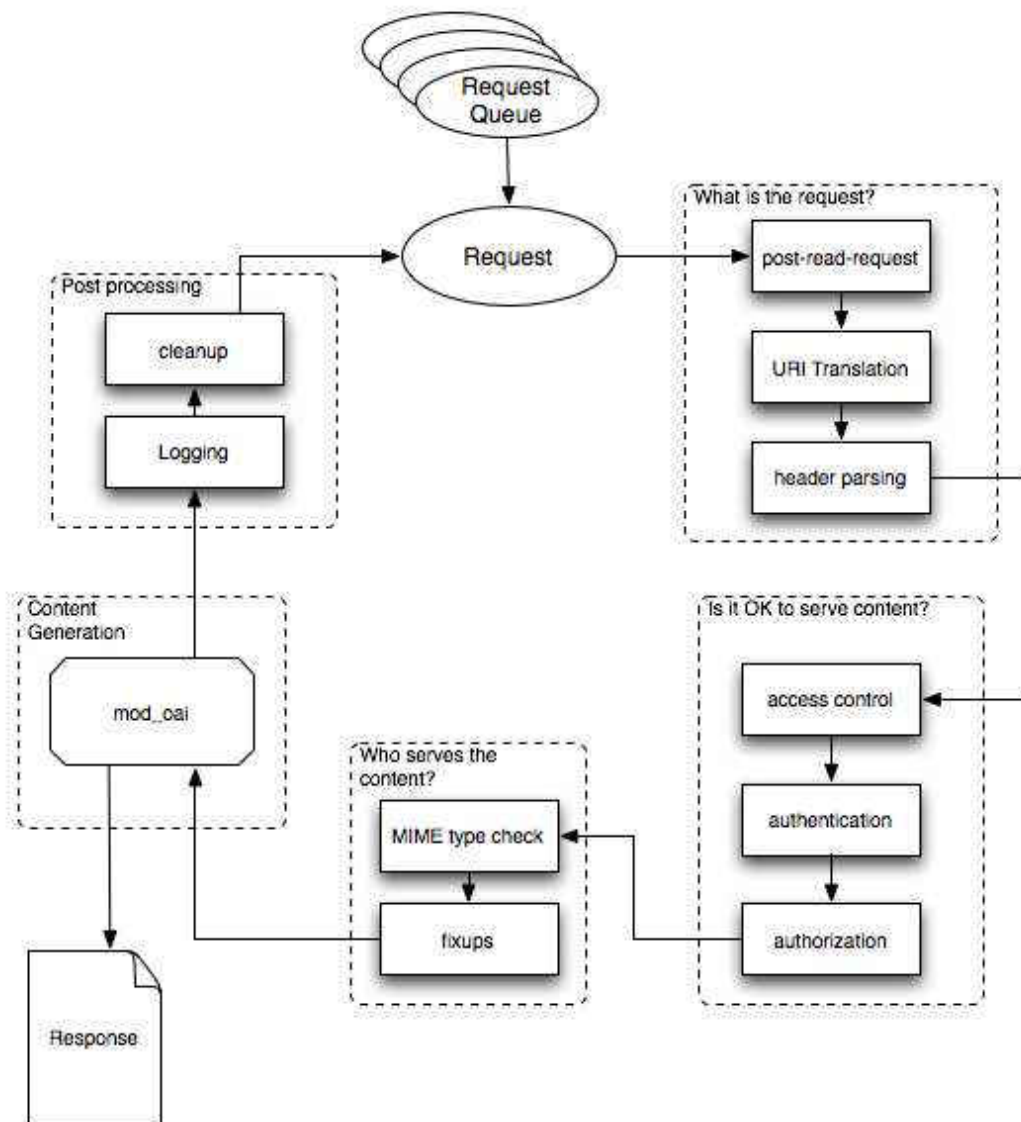


FIG. 59: The Apache web server request processing loop.

the list of parameters associated with a resumption token. For the former, the key fields are verb, identifier, metadataPrefix, from, until, set and resumptionToken, as specified in the OAI-PMH specification. For the latter case, the keys are the tokens used to save a clients state: firstUnseenRecord, metadataPrefix, from, until and set. Using apr_tables for these lists provides a easy, platform independent data structure for managing this data.

apr_xml_parser The XML Parser function provides platform independent DOM based parsing. The MODDOI 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 MODDOI, 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 MODDOI code from tedious memory management tasks, and ensures leak free memory management.

2 HISTORY OF MODDOI

The original MODDOI software was developed by several graduate students at Old Dominion University between 2000 and 2003. The author took over maintenance of the codebase in 2003, and initially focused on segmenting the code into a more maintainable structure. The code was refactored from a single file containing all the source code into eighteen source code files organized by structural areas.

When taking over existing code, there is a danger of loosing existing functionality because of lack of understanding of the more subtle aspects the 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 testing of many testcases 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 MODDOI 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. The configurable metadata extraction utilities were handled with the new plugin architecture, while the built-in metadata extraction utilities (HTTP-HEADER, OAI-DC etc) were handled in the old, monolithic fashion.

This inconsistency was resolved by refactoring the legacy built-in utilities to be internally defined as plugins. Internally, they differ from the new utilities only in that they are not defined in

the configuration file, and that they do not depend on external utilities. They are completely implemented within the MODOAI program itself. This refactoring greatly simplifies the code, allowing reduction of duplicate functionality.

Next, the code was extended to specify the list of resources in the site through a sitemap file, rather than by filesystem traversal at dissemination time. This 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 filesystem traversal mechanism to select the exposed resources. Short of manual examination of the MODOAI source code or manual sampling of MODOAI query results, webmasters had no visibility into the exposed resource list.

Second, reading a single sitemap file has better performance than manually traversing the filesystem to generate the resource list. The traversal happens once, at sitemap generation time, rather than each time a MODOAI request is serviced.

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.

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.

The above refactoring allowed the author to rework resumption token support. Resumption token support is now centralized. Immediately after parsing the sitemap file, the list of resource is pruned based on the `modoai_max_response_items` 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.

3 THE DESIGN & STRUCTURE OF MODOAI

The MODOAI module is written in ANSI C, with external dependencies on the Apache HTTP server and the Apache Portable Runtime.

The dependencies to APR are resolved through the APache eXtenSion (apxs) tool. apxs is compiled with and distributed as a part of the Apache HTTP server. It exposes knowledge about the compile time settings and file system paths used by the Apache HTTP server. For example, to resolve the location of the installed Apache HTTP include files, the MODOAI Makefile queries apxs (indirectly) using the following command line:

```
> apxs -q INCLUDEDIR
/usr/include/apache2
```

Structurally, the MODOAI codebase is segregated into 18 ANSI C files, with 18 corresponding header files publishing the public interfaces to MODOAI functions.¹

error.c Contains code for generating OAI-PMH error responses.

grammar.c Contains code for checking the grammar of incoming request. This ensures that MODOAI does not attempt to handle any incoming request which does not conform to valid OAI-PMH grammar.

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.

```
vGetRecord.c    vListSets.c    vListMetadataFormats.c
vIdentify.c    vListRecords.c    vListIdentifiers.c]
```

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

¹All moduai responses are HTTP response code 200 because they return a response from moduai as opposed to failing at the HTTP level.

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 external plugin utilities.

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

timefunc.c Date and time formatting utilities.

3.1 Module Configuration

The lifetime of an Apache HTTP server is depicted in Figure 59 on page 126. At startup, Apache parses the config 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 hooks into the Apache life cycle: per-server-config, per-directory-config and 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. See lines 11–21 of Appendix D–1 on page 178. At startup, this parsing only reads the files for correct syntax, but 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 is configured to pass on to MODOAI.

3.2 Request Handling

Apache executes the MODOAI callback `modoai2_method_handler()` when an OAI-PMH request arrives. Figure 60 on the following page depicts the internal processing MODOAI follows to handle the request. The MODOAI module handles the response in seven stages.

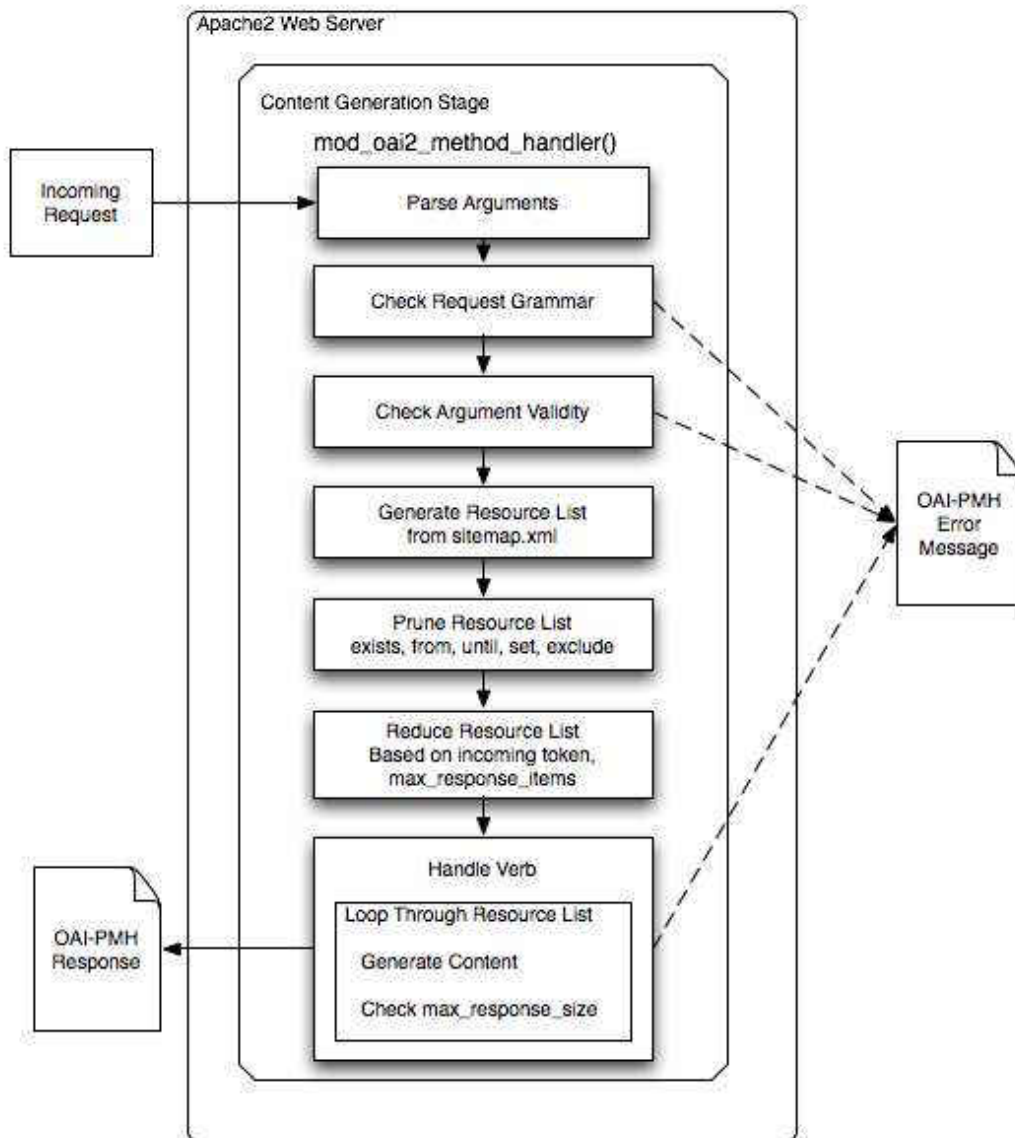


FIG. 60: The MOD_OAI processing loop.

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 they 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.

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 ListIdentifier verbs. To present a request with a resumption token and an Identify verb is not valid. The MODOAI module will issue an error page and stop further processing.

All OAI-PMH requests except Identify and ListMetadataFormats 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 filesystem are removed Any resources which are excluded by a from, until, or set in the query are removed Any resources covered by an modoi_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 modoi_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 7: The final stage is to actually process the request. For each resource, output XML is generated, and a running total of the number of characters generated is tallied. If after processing a resource, the total is compared with the configuration parameter modoi_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.

4 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`² file, through the `modoi_plugin` directive. The plugin can be any program which can be executed from the command line. The configuration directive, an example of which is below, takes four arguments.

```
modoi_plugin zipinfo "/usr/bin/zipinfo %s"
"/usr/bin/zipinfo -version" */zip
```

The first argument is a label which is used to identify the plugin within the output XML file. The second argument is a command line to execute to generate the plugin output. The command line is parsed for the string `%s`, which is replaced with the name of the filesystem path to the resource being examined. The third argument also a command line program to execute, but 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` can be provided to give an appropriate version command. 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 `exiftool` utility examines metadata for digital photography files. It would not be appropriate to run this against a file with a MIME-type of `text/html`. For `exiftool`, a configuration could be created which only matches `image/*`, which would match both `image/jpeg` and `image/raw`.

In some cases, wrapper scripts are needed to allow utilities to conform to the above convention. One such example is the `Jhove` utility. The `Jhove` command line requires a specification of the `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 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 %s`. To use `Jhove` for only JPEG images, the MODOAI configuration line should appear as:

```
modoi_plugin jhove "/opt/jhove/procJhove.pl %s" "/opt/jhove/jhove
-c /opt/jhove/conf/jhove.conf -h xml -v" "image/jpeg"
```

The MODOAI plugin architecture is designed to make it easy to add additional metadata extraction utilities, including other `Jhove` lines to have one for each type of hul processor desired.

²Some versions of Apache use a single `httpd.conf` file. Others use a combination of files, of which `modoi.conf` would be one.

5 CRATE DEPLOYED IN MODOAI

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 familiar to most webmasters, it does not impose significant additional workload. The primary tool which enables CRATE is just another in a long list of Apache web server modules (MODOAI) 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. Another advantage in using MODOAI is that webmasters are familiar with their local resource types and can select plugin metadata extractors that are appropriate to the site. The *feasibility* of such an implementation was demonstrated in a series of experiments which are described in the next section.

6 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 reworkedMODOAI module now treats built-in metadata extraction utilities in the same manner as webmaster-defined metadata extraction utilities, simplifying code maintenance. The author has refactored the original code, and extended it to implement an extensible plugin architecture. New metadata utilities can be added by webmasters through configuration in the standard Apache httpd.conf configuration file. Code regressions are avoided by execution of a new functional test suite prior to all code commits.

CHAPTER XI

CONCLUSIONS

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 [147]

1 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.

1.1 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 [152, 151], 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.

1.2 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 web server 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 MOD_OAI has been released to the public under a GPL-2 license and is available through Google Code (http://code.google.com/mod_oai) and at the MOD_OAI website (http://www.mod_oai.org/).

2 FUTURE WORK

This dissertation presents a number of accomplishments with regard to integrating preservation functionality into the web server. More remains to be done, both in the realm of the CRATE Model and in software engineering.

2.1 General Investigations

1. The CRATE Model describes an XML document output in plain ASCII. As UTF-8 becomes the character-encoding of choice in the international community, the use of UTF-8 instead of ASCII should be investigated. Since UTF-8 is backward-compatible with ASCII, a transition will not adversely affect earlier installations where ASCII was employed. For multi-character-set compatibility, the Unicode standard and its most prevalent implementation, UTF-8, should be considered as the future basis for CRATE objects. Other encodings may become prevalent or prove useful.
2. OAI-ORE resource maps should be investigated for CRATE compatibility.
3. There are other non-HTTP protocols that could participate in preservation such as the RSS and ATOM syndication formats.

2.2 Web Server-Related Investigations

1. One feature of HTTP is the “Accept-Encoding” field. A CRATE-specific encoding could be developed which would indicate to the server that the client is requesting the response to consist of the resource in CRATE Model format.
2. 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.
3. In addition to general website testing, investigations into the impact on virtual hosting environments (many webs, 1 machine) may provide further insights into feasibility for such sites and recommendations for improving performance.

4. Most metadata utilities that were available for use during this research were not developed with this implementation in mind. Further investigations into optimization of metadata utilities would be helpful if web servers are to process resources for preservation metadata.
5. The Model should be implemented in other web server environments (Microsoft's IIS for example) to see if it is practical for these servers.
6. Proliferation is an important component of success for this Model. 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.

2.3 Further MODOAI Development

1. The Apache API has certain limitations that impact the ability of modules to access dynamic content. Use of undocumented features can produce problems in future versions of Apache, so the current release of MODOAI limits itself to elements for which it is the registered handler. As a result, it cannot do speculative evaluation of, for example, dynamic content for which the registered handler is the content-generating module. Alternate solutions for this problem should be investigated.
2. 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.
3. 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.

3 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 web sites, 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. 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. Websites would benefit from the democratization of preservation functionality in this nearly-transparent, server-implemented solution. The CRATE Model is the key enabling technology in making this transformation possible.

BIBLIOGRAPHY

- [1] Amazon, Inc. Kindle official site. <http://kindle.amazon.com/>.
- [2] Archive it. <http://www.archive-it.org/>.
- [3] William Y. Arms. Key concepts in the architecture of the digital library. *D-Lib Magazine*, 1, July 1995. <http://www.dlib.org/dlib/July95/07arms.html>.
- [4] Auditmypc.com. Webmaster tool. Java servlet for sitemap generation, 2008. <http://www.auditmypc.com/xml-sitemap.asp>.
- [5] Autositemap.com. Online sitemap tool, 2008. <http://www.autositemap.com/>.
- [6] Ricardo Baeza-Yates, Carlos Castillo, and Efthimis N. Efthimiadis. Characterization of national web domains. *ACM Transactions on Internet Technology TOIT*, 7(2), 2007.
- [7] Ziv Bar-Yossef, Idit Keidar, and Uri Schonfeld. Do not crawl in the DUST: Different URLs with Similar Text. In *Proceedings of the 16th International World Wide Web Conference WWW'07*, pages 111–120, 2007.
- [8] Jeroen Bekaert, Emiel De Kooning, and Herbert Van de Sompel. 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.
- [9] Jeroen Bekaert and Herbert Van de Sompel. A standards-based solution for the accurate transfer of digital assets. *D-Lib Magazine*, 11(6), June 2005. doi:10.1045/june2005-bekaert.
- [10] Leeann Bent, Michael Rabinovich, Geoffrey M. Voelker, and Zhen Xiao. 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*, volume 9, pages 522–533. ACM, December 2004. <http://doi.acm.org/10.1145/988672.988743>.
- [11] Michael K. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1), 2001.
- [12] K. Bharat and A. Broder. Mirror, mirror on the Web: A study of host pairs with replicated content. *WWW8 Computer Networks*, 31(11-16):1579–1590, 1999.
- [13] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The harvest information discovery and access system. In *Proceedings of the Second International*

WWW Conference '94: Mosaic and the Web, October 1994. <http://portal.acm.org/citation.cfm?coll=GUIDE\&dl=GUIDE\&id=217463>.

- [14] C.M. Bowman, P.B. Danzig, D.R. Hardy, U. Manber, and M.F. Schwartz. The harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28(1-2):119–125, 1995.
- [15] Stewart Brand. Far forward. Report from the RLG OCLC Annual Meeting, 2000. <http://digitalarchive.oclc.org/da/ViewObjectMain.jsp;jsessionid=84ae0c5f82405355a4cd14ef4b89b948a6af86f3bee0?fileid=0000070504:000006276583&reqid=7>.
- [16] Onn Brandman, Junghoo Cho, Hector Garcia-Molina, and Narayanan Shivakumar. Crawler-friendly web servers. *SIGMETRICS Perform. Eval. Rev.*, 28(2):9–14, 2000. <http://doi.acm.org/10.1145/362883.362894>.
- [17] William J. Broad. Web archive is said to reveal a nuclear primer. *The New York Times Online*, 3 November 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/9p10jiGDw&oref=slogin&oref=slogin.
- [18] Carlos Castillo. Cooperation schemes between a web server and a web search engine. In *LA-WEB*, pages 212–213. IEEE CS Press, 2003. citeseer.ist.psu.edu/castillo03cooperation.html.
- [19] James Caverlee, Ling Liu, and David Buttler. Probe, cluster, and discover: Focused extraction of qa-pagelets from the deep web. In *ICDE*, pages 103–115, 2004. <http://csdl.computer.org/comp/proceedings/icde/2004/2065/00/20650103abs.htm>.
- [20] CCSDS. Reference model for an open archival information system ISO 14721:2002. Technical Report CCSDS 650.0-B-1, Consultative Committee for Space Data Systems, January 2002.
- [21] Lois Mai Chan. Inter-indexer consistency in subject cataloging. *Information Technology and Libraries*, 8(4):349–358, December 1989.
- [22] Mo Chen, Jian-Tao Sun, Hua-Jun Zeng, and Kwok-Yan Lam. A practical system of keyphrase extraction for web pages. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 277–278. ACM, 2005. <http://doi.acm.org/10.1145/1099554.1099625>.

- [23] Ludmila Cherkasova and Magnus Karlsson. Dynamics and evolution of web sites: Analysis, metrics, and design issues. In *ISCC*, pages 64–71. IEEE Computer Society, July 2001.
- [24] Junghoo Cho and Hector Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Trans. Database Syst.*, 28(4):390–426, 2003. <http://doi.acm.org/10.1145/958942.958945>.
- [25] Junghoo Cho and Hector Garcia-Molina. Estimating frequency of change. *ACM Transactions on Internet Technology*, 3(3):256–290, 2003. <http://doi.acm.org/10.1145/857166.857170>.
- [26] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. 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](http://dx.doi.org/10.1016/S0169-7552(98)00108-1).
- [27] Junghoo Cho, Narayanan Shivakumar, and Hector Garcia-Molina. Finding replicated web collections. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 355–366. ACM Press, 2000. <http://doi.acm.org/10.1145/342009.335429>.
- [28] 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)*, page 26. National Academies Press, 1995. <http://darwin.nap.edu/books/NI000157/html/26.html>.
- [29] Connexion Integrated Cataloging Service. OCLC, Online Computer Library Center. <http://www.oclc.org/connexion/>.
- [30] DASL: DAV Searching and Locating. <http://www.webdav.org/dasl/> (accessed on 3/17/2006).
- [31] The Dublin Core Metadata Initiative. History of the Dublin Core Metadata Initiative. <http://dublincore.org/about/history/>.
- [32] DVDXCOPY. web site. <http://www.dvdxcopy.com/default.asp>.
- [33] European archive. <http://www.europarchive.org/>.
- [34] Jerry Everard. Meta tags – so 200BCE, March 2007. <http://lostbiro.com/blog/wp-content/uploads/2007/03/RosettaStone.jpg>.
- [35] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet L. Wiener. A large-scale study of the evolution of web pages. *Software: Practice & Experience*, 34(2):213–237, 2004.

- [36] flickr. Yahoo's flickr site; a subcategory of images with OAI-PMH tags, 2006. <http://flickr.com/photos/tags/oaipmh/>.
- [37] N. Freed and N. Borenstein. Multipurpose internet mail extensions (MIME) Part One: Format of Internet Message Bodies. IETF RFC 2045, November 1996. <http://tools.ietf.org/html/rfc2045>.
- [38] N. Freed and N. Borenstein. Multipurpose internet mail extensions (MIME) Part Two: Media Types. IETF RFC 2046, November 1996. <http://tools.ietf.org/html/rfc2046>.
- [39] Frontiernet website. NIST Logo conversion, May 2008. http://www.frontiernet.net/~imaging/sc_nist_logo_java3d.jpg.
- [40] H. M. Gladney. Trustworthy 100-year digital objects: Evidence after every witness is dead. *ACM Transactions On Information Systems*, 22(3):406–436, July 2004.
- [41] Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen. HTTP extensions for distributed authoring – WEBDAV. Technical Report Internet RFC-2518, IETF, 1999.
- [42] Google, Inc. 20 year archive on google groups. http://www.google.com/googlegroups/archive_announce_20.html.
- [43] Google, Inc. First mention of MS-DOS. Posting by "Frank", June 1981. <http://groups.google.com/group/fa.info-cpm/msg/c86239309bfbd31>.
- [44] Google, Inc. World Wide Web - Executive Summary. Posting by Tim Berners-Lee, August 1991. <http://groups.google.com/group/alt.hypertext/msg/395f282a67a1916c>.
- [45] Google, Inc. Creating google sitemaps files. <http://www.google.com/support/webmasters/bin/topic.py?topic=8467>, 2006.
- [46] Google, Inc. Google code: Web authoring statistics. <http://code.google.com/webstats/>, 2006. Accessed on 10 Feb 2007.
- [47] Google, Inc. Sitemap tool. Python tool for sitemap generation, 2008. <https://www.google.com/webmasters/tools/docs/en/sitemap-generator.html>.
- [48] Google, Inc. Webmaster guidelines, February 2008. <http://www.google.com/support/webmasters/bin/answer.py?answer=35769>.
- [49] James V. Grimaldi, Juliet Eilperin, and Jonathan Weisman. Some say they felt uneasy about representative's attention. *The Washington Post*, page Page A01, 4 October 2006.

- [50] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. *WWW 2005*, 5, May 2005.
- [51] Darren R. Hardy and Michael F. Schwartz. 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>.
- [52] Henry A. Murray Research Archive. The Institute for Quantitative Social Science at Harvard University. <http://www.murray.harvard.edu/frontpage>.
- [53] Harvest: A distributed search system. Sourceforge Project. <http://harvest.sourceforge.net/>.
- [54] Michael Hauben and Ronda Hauben. *Netizens: On the History and Impact of Usenet and the Internet*. Wiley IEEE Computer Society Press, May 1997. ISBN 0-8186-7706-6.
- [55] Kevin Hemenway and Tara Calishain. *Spidering Hacks*. O’Reilly Media, Inc., first edition, November 2003.
- [56] M.R. Henzinger. Hyperlink analysis for the Web. *IEEE Internet Computing*, 5(1):45–50, Jan/Feb 2001.
- [57] Allan Heydon and Marc Najork. Mercator: A scalable, extensible web crawler. In *WWW ’99: Proceedings of the 8th International Conference on World Wide Web*, pages 219–229, Toronto, Canada, 1999. <http://dx.doi.org/10/1023/A:1019213109274>.
- [58] Jun Hirai, Sriram Raghavan, Hector Garcia-Molina, and Andreas Paepcke. WebBase: a repository of web pages. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):277–293, 2000. citeseer.ist.psu.edu/article/hirai99webbase.html.
- [59] International Conference on Preservation of Digital Objects. <http://ipres.library.cornell.edu/>.
- [60] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: Categorizing hidden web databases. In *SIGMOD Conference*, pages 100–109, 2001. <http://doi.acm.org/10.1145/375663/375671>.
- [61] S. Jamin, C. Jin, A.R. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placement on the Internet. *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, 1, 2001.
- [62] Robert Kahn and Robert Wilensky. A framework for distributed digital object services. Corporation for National Research Initiatives hdl:cnri.dlib/tn95-01, May 1995. <http://www.cnri.reston.va.us/home/cstr/arch/k-w.html>.

- [63] Robert Kahn and Robert Wilensky. A framework for distributed digital object services. *International Journal On Digital Libraries*, 6(2):115–123, April 2006.
- [64] KEA automatic keyphrase extraction tool. New Zealand Digital Library, University of Waikato, 2006. <http://www.nzdl.org/Kea/>.
- [65] Martin Klein, Frank McCown, Joan A. Smith, and Michael L. Nelson. *How Much Preservation Do I Get If I Do Absolutely Nothing?*, pages 71 – 87. GITO-Verlag, Berlin, Germany, 2007.
- [66] Koninklijke bibliotheek. (National Library of the Netherlands). <http://www.kb.nl/>.
- [67] Carl Lagoze, Herbert Van de Sompel, Michael L. Nelson, and Simeon Warner. The Open Archives Initiative Protocol for Metadata Harvesting. <http://www.openarchives.org/OAI/openarchivesprotocol.html>.
- [68] Carl Lagoze and Herbert Van de Sompel. 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*, pages 54–62, New York, NY, USA, 2001. ACM Press. <http://doi.acm.org/10.1145/379437.379449>.
- [69] Carl Lagoze, Herbert Van de Sompel, Michael L. Nelson, and Simeon Warner. Implementation guidelines for the Open Archives Initiative Protocol for Metadata Harvesting. <http://www.openarchives.org/OAI/2.0/guidelines.htm>, 2005.
- [70] Lawrence E. Leonard. 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.
- [71] Ryan Levering and Michal Cutler. The portrait of a common html web page. In *DocEng '06: Proceedings of the 2006 ACM symposium on Document engineering*, pages 198–204, New York, NY, USA, October 2006. ACM. <http://doi.acm.org/10.1145/1166160.1166213>.
- [72] David M. Levy. Heroic measures: reflections on the possibility and purpose of digital preservation. In *DL '98: Proceedings of the Third ACM Conference on Digital Libraries*, pages 152–161, New York, NY, USA, 1998. ACM Press. <http://doi.acm.org/10.1145/276675.276692>.
- [73] Library of Congress. *The State of Digital Preservation: An International Perspective*, Washington, DC, July 2002. Council on Library Information Resources (CLIR).

- [74] Xiaoming Liu, Lyudmila Balakireva, and Patrick Hochstenbach and Herbert Van de Sompel. File-based storage of digital objects and constituent datastreams: XMLtapes and internet archive ARC files. In *9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2005)*, pages 254–265, Sept 2005.
- [75] Zhenyu Liu, Chang Luo, Junghoo Cho, and Wesley W. Chu. A probabilistic approach to metasearching with adaptive probing. In *ICDE*, pages 547–559, 2004. <http://doi.ieeecomputersociety.org/10.1109/ICDE.2004.1320026>.
- [76] Peter Lyman. Archiving the world wide web. In *Preserving Our Digital Heritage: Plan for the National Digital Information Infrastructure and Preservation Program*, page Appendix 2. Library of Congress, October 2002. http://www.digitalpreservation.gov/about/es_web.pdf.
- [77] Peter Lyman. Archiving the world wide web. In *Building a National Strategy for Preservation: Issues in Digital Media Archiving*. Council on Library and Information Resources, 2002.
- [78] Peter Lyman, Hal R. Varian, Peter Charles, Nathan Good, Laheem Lamar Jordan, and Joyojeet Pal. How much information? 2003. Research Project Report, U.C. Berkeley School of Information Management and Systems, October 2003. <http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/>.
- [79] Clifford Lynch. Metadata harvesting and the Open Archives Initiative. ARL Bi-Monthly Report, August 2001. <http://www.arl.org/newsltr/217/mhp.html>.
- [80] Clifford Lynch. 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](http://dx.doi.org/10.1002/1532-2890(2000)52:1<12::AID-ASI1062>3.3.CO;2-M).
- [81] Kurt Maly, Michael L. Nelson, and Mohammed Zubair. Smart objects, dumb archives. *D-Lib Magazine*, March 1999. <http://www.dlib.org/dlib/march99/maly/03maly.html>.
- [82] Petros Maniatis, Mema Roussopoulos, T.J.Giuli, David S. H. Rosenthal, and Mary Baker. The LOCKSS peer-to-peer digital preservation system. *ACM Transactions on computer systems*, 23(1):2 – 50, February 2005.
- [83] Frank McCown, Amine Benjelloun, and Michael L. Nelson. Brass: A queueing manager for warrick. In *7th International Web Archiving Workshop (IWA 2007)*, June 2007.

- [84] Frank McCown, Norou Diawara, and Michael L. Nelson. Factors affecting website reconstruction from the web infrastructure. In *JCDL'07: Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries*, pages 39 – 48, June 2007. doi:10.1145/1255175.1255182.
- [85] Frank McCown, Xiaoming Liu, Michael L. Nelson, and Mohammed Zubair. Search engine coverage of the OAI-PMH corpus. *IEEE Internet Computing*, 10(2):66–73, March/April 2006.
- [86] Frank McCown, Joan A. Smith, Michael L. Nelson, and Johan Bollen. Reconstructing websites for the lazy webmaster. Technical Report arXiv:cs.IR/0512069, Old Dominion University, 2005. <http://arxiv.org/abs/cs.IR/0512069>.
- [87] Frank McCown, Joan A. Smith, Michael L. Nelson, and Johan Bollen. Reconstructing websites for the lazy webmaster. *Proceedings of the eighth ACM international workshop on web information and data management (WIDM)*, pages 67–74, November 2006. <http://doi.acm.org/10.1145/1183550.1183564>.
- [88] Jerome P. McDonough. METS: Standardized encoding for digital library objects. *International Journal on Digital Libraries*, 6(2):148–158, April 2006. 10.1007/s00799-005-0132-1.
- [89] Eric Miller. An introduction to the resource description framework. *D-Lib Magazine*, 4(5), 1998. <http://dlib.org/dlib/may98/miller/05miller.html>.
- [90] Gilad Mishne, David Carmel, and Ronny Lempel. Blocking blog spam with language model disagreement. In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web AIR Web '05*, May 2005.
- [91] G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic. Introduction to heritrix, an archival quality web crawler. In *Proceedings of the 4th International Web Archiving Workshop (IWA'04)*, Sept 2004.
- [92] John Mueller. Retiring support for OAI-PMH in sitemaps. Google Webmaster Central Blog, April 2008. <http://googlewebmastercentral.blogspot.com/2008/04/retiring-support-for-oai-pmh-in.html>.
- [93] The national archives digital preservation department (UK). <http://www.nationalarchives.gov.uk/preservation/digital.htm>.
- [94] National Library of Australia Digital Services Project. <http://www.nla.gov.au/nla/staffpaper/dw001004.html>.

- [95] NDIIPP, the National Digital Information Infrastructure and Preservation Program. <http://www.digitalpreservation.gov/>.
- [96] Michael L. Nelson, Brad Argue, Miles Efron, Sheila Denn, and Maria Cristina Pattuelli. A survey of complex object technologies for digital libraries. Technical Report NASA/TM-2001-211426, NASA Langley Research Center, 2001. <http://techreports.larc.nasa.gov/ltrs/PDF/2001/tm/NASA-2001-tm211426.pdf>.
- [97] Michael L. Nelson, Johan Bollen, Giridhar Manepalli, and Rabia Haq. Archive ingest and handling test, the Old Dominion University approach. *D-Lib Magazine*, 11(12), October 2005. doi:10.1045/december2005-nelson.
- [98] Michael L. Nelson, Frank McCown, Joan A. Smith, and Martin Klein. Using the web infrastructure to preserve web pages. *International Journal on Digital Libraries*, 6(4):327–349, April 2007. doi:10.1007/s00799-007-0012-y.
- [99] Michael L. Nelson, Joan A. Smith, Herbert Van de Sompel, Xiaoming Liu, and Ignacio Garcia del Campo. Efficient, automatic web resource harvesting. In *Proceedings of the seventh ACM international workshop on web information and data management*, pages 43–50, November 2006. <http://doi.acm.org/10.1145/1183550.1183560>.
- [100] Michael L. Nelson and Herbert Van de Sompel. 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.
- [101] Netcraft server survey. http://news.netcraft.com/archives/web_server_survey.html.
- [102] Alexandros Ntoulas, Junghoo Cho, and Christopher Olston. 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*, pages 1–12, 2004. <http://doi.acm.org/10.1145/988672.988674>.
- [103] Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. Detecting spam web pages through content analysis. In *WWW'06: Proceedings of the 15th International Conference on World Wide Web*, 2006.
- [104] Alexandros Ntoulas, Petros Zerfos, and Junghoo Cho. Downloading textual hidden web content through keyword queries. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 100–109, 2005. <http://doi.acm.org/10.1145/1065385.1065407>.

- [105] Edward T. O'Neill, Brian F. Lavoie, and Rick Bennett. Trends in the evolution of the public web. *D-Lib Magazine*, 3(4), April 2003. <http://dx.doi.org/10.1045/april2003-lavoie>.
- [106] G. Pant, P. Srinivasan, and F. Menczer. *Crawling the Web*, chapter II, pages 153 – 178. Springer-Verlag, 2004.
- [107] Juan Carlos Perez. Google, Yahoo and Microsoft partner to help webmasters. *Computerworld*, November 2006. <http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9005148>.
- [108] Mark Pilgrim. What is rss. *O'Reilly XML.com*, 2002. <http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html>.
- [109] Gary Price. Web search engines FAQs: questions, answers, and issues. *Searcher*, 9(9), October 2001. <http://www.infotoday.comc/searcheroct01/price.htm>.
- [110] Sriram Raghavan and Hector Garcia-Molina. Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 129–138, 2001. <http://dbpubs.stanford.edu/pub/2000-25>.
- [111] Resource description framework. W3C Recommendation. <http://www.w3.org/RDF/>.
- [112] J. Reagle. Web RSS (syndication) history, 2003. <http://goatee.net/2003/rss-history.html>.
- [113] G.S. Robinson and C. Cargill. History and impact of computer standards. *Computer*, 29(10):79 – 85, October 1996.
- [114] D. Rosenthal, T. Robertson, T. Lipkis, T. Reich, and S. Morabito. Requirements for digital preservation systems. a bottom-up approach. *D-Lib Magazine*, 11(11), 2005. doi:10.1045/november2005-rosenthal.
- [115] The rosetta stone. Online photo by Eduardo Rocha, PhD. http://erocha.freehosting.net/Rosetta_Stone_WEB2.jpg.
- [116] Rosetta stone replica. From the Museum Store Company online catalog, 2008. <http://museumstorecompany.com/index.php?cPath=10>.
- [117] Jeff Rothenberg. Ensuring the longevity of digital information. *Scientific American*, 272(1):42–47, January 1995.
- [118] Jeff Rothenberg. Ensuring the longevity of digital information. <http://www.clir.org/pubs/archives/ensuring.pdf>, February 1999. Revision 980327.

- [119] Mike Scott. Wordsmith software package. Oxford University Press, 2008. <http://www.lexically.net/wordsmith/>.
- [120] Chris Sherman. Getting the New York Times more search engine friendly. *Search Engine Watch*, June 15 2006. <http://searchenginewatch.com/showPage.html?page=3613561>.
- [121] Clay Shirky. Ontology is overrated: Categories, links, and tags. Written synthesis of two conference talks. http://www.shirky.com/writings/ontology_overrated.html.
- [122] Clay Shirky. AIHT Conceptual issues from practical tests. *D-Lib Magazine*, 11(12), December 2005. <http://www.dlib.org/dlib/december05/shirky/12shirky.html>.
- [123] Clay Shirky. Library of Congress Archive Ingest and Handling Test AIHT final report. Report by the National Digital Information Infrastructure & Preservation Program, June 2005.
- [124] Jol A. Silverman. Photographic evidence, naked children, and dead celebrities: Digital forgery and the law, April 1998. <http://www.thirdamendment.com/digital.html>.
- [125] Simon Singh. *The Codebook: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchorbooks, September 2000.
- [126] Sitemaps XML Format. Sitemaps Protocol, 2008. <http://www.sitemaps.org/protocol.php>.
- [127] Joan Smith and Michael Nelson. Using OAI-PMH resource harvesting and MPEG-21 DIDL for digital preservation. 2nd International Conference on Open Repositories 2007, January 2007.
- [128] Joan A. Smith, Frank McCown, and Michael L. Nelson. Observed web robot behavior on decaying web subsites. *D-Lib Magazine*, Feb 2006. <http://www.dlib.org/dlib/february06/smith/02smith.html>.
- [129] Joan A. Smith and Michael L. Nelson. Generating best-effort preservation metadata for web resources at time of dissemination. In *Proceedings of the Joint Conference on Digital Libraries(JCDL 2007)*, pages 51–52, June 2007.
- [130] Joan A. Smith and Michael L. Nelson. Creating preservation-ready web resources. *D-Lib Magazine*, 14(1/2), 2008. doi:10.1045/january2008-smith.

- [131] Joan A. Smith and Michael L. Nelson. 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.
- [132] The Unica Island Team. Rosetta stone, 1997. <http://library.thinkquest.org/10005/media/photos/RosettaStone.gif>.
- [133] The Apache Software Foundation. Apache HTTP server version 2.0 documentation, 2008. <http://httpd.apache.org/docs/2.0/>.
- [134] Charles F. Thomas and Linda S. Griffin. Who will Create The Metadata For the Internet? *First Monday*, 3(12), 1998. http://www.firstmonday.dk/issues/issue3_12/thomas/.
- [135] Herbert Van de Sompel and Carl Lagoze. 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>.
- [136] Herbert Van de Sompel, Michael L. Nelson, Carl Lagoze, and Simeon Warner. Resource harvesting within the OAI-PMH framework. *D-Lib Magazine*, 10(12), December 2004. doi:10.1045/december2004-vandesompel.
- [137] Herbert Van de Sompel, Jeffery A. Young, and Thomas B. Hickey. Using the OAI-PMH ... differently. *D-Lib Magazine*, 9(7/8), 2003. <http://dx.doi.org/10.1045/july2003-young>.
- [138] Arthur van Hoff, John Giannandrea, Mark Hapner, Steve Carter, and Milo Medin. The HTTP distribution and replication protocol, August 1997. <http://www.w3.org/TR/NOTE-drp>.
- [139] 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.
- [140] *Web Characterization Terminology and Definition Sheet, W3C Working Draft 24-May-1999*, May 1999. <http://www.w3.org/1999/05/WCA-terms/>.
- [141] W3C. RDFa in XHTML: syntax and processing. Technical report, W3C, February 2008. <http://www.w3.org/TR/2008/WD-rdfa-syntax-20080221/>.
- [142] WARP: National Diet Library of Japan, Web Archiving Project. (A summary report in English). http://www.ndl.go.jp/en/iflapac/e_resources.html.

- [143] Andrew Waugh. The design of the VERS encapsulated object experience with an archival information package. *International Journal on Digital Libraries*, 6(2):184–191, April 2006.
- [144] Andrew Waugh. The design and implementation of an ingest function to a digital archive. *D-Lib Magazine*, November/December 2007. doi:10.1045/november2007-waugh.
- [145] Andrew Waugh, Ross Wilkinson, Brendan Hills, and Jon Dell’oro. Preserving digital information forever. In *DL ’00: Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 175–184, 2000. <http://doi.acm.org/10.1145/336597.336659>.
- [146] The Wayback Machine Frequently Asked Questions. <http://www.archive.org/about/faqs.php> (Accessed 18 January 2006).
- [147] Colin Webb. National Library of Australia preservation metadata for digital collections. <http://www.nla.gov.au/preserve/pmeta.html>, October 1999.
- [148] Stuart Weibel. Metadata: The foundations of resource description. *D-Lib Magazine*, 1(1), 1995. <http://dx.doi.org/10.1045/july95-weibel>.
- [149] 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.
- [150] XMLSITEMAP.COM. Xmlsitemap. Web-based sitemap tool., 2008. <http://xmlsitemap.com/create-sitemap/>.
- [151] Phil Zimmerman. *The Official PGP User’s Guide*. MIT Press, Cambridge, MA, May 1995.
- [152] Phil Zimmerman. Why I wrote PGP, 1999. <http://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html>.

APPENDIX A

CRATE XML SCHEMA DOCUMENTS

1 THE SIMPLE CRATE SCHEMA: CRATE.XSD

```

1 <?xml version="1.0"?>
2 <xsd:annotation>
3 <xsd:documentation>
4   This is a schema to represent the CRATE model.
5   Schema author: Joan A. Smith
6 </xsd:documentation>
7 </xsd:annotation>
8 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9   xmlns="http://www.cratemodel.org/xmlns/"
10  elementFormDefault="qualified">
11 <xsd:element name="Crate" minOccurs="1" maxOccurs="unbounded">
12 <xsd:complexType>
13 <xsd:sequence>
14 <xsd:element name="ResourceID" type="xsd:string"/>
15 <xsd:element name="MetadataUtility"
16 type="MetadataUtilityType"
17   minOccurs="1" maxOccurs="unbounded"/>
18 <xsd:element name="Resource"
19   type="ResourceType" minOccurs="1">
20 </xsd:sequence>
21 </xsd:complexType>
22 </xsd:element>
23 <xsd:complexType name="MetadataUtilityType">
24 <xsd:sequence>
25 <xsd:element name="Name" type="xsd:string"
26   minOccurs="1" maxOccurs="1"/>
27 <xsd:element name="Exec" type="xsd:string"
28   minOccurs="0" maxOccurs="1"/>
29 <xsd:element name="Version" type="xsd:string"
30   minOccurs="1" maxOccurs="1"/>
31 <xsd:element name="MimeType" type="xsd:string"
32   minOccurs="0" maxOccurs="1"/>
33 <xsd:element name="ExecTimeStamp" type="xsd:timeInstant"
34   minOccurs="0"/>
35 <xsd:element name="ResourceMetadata" type="xsd:string"
36   minOccurs="0" maxOccurs="1"/>
37 <xsd:any namespace="http://www.cratemodel.org/xmlns/"
38   minOccurs="0"/>
39 </xsd:sequence>
40 </xsd:complexType>
41 <xsd:complexType name="ResourceType">
42 <xsd:sequence>
43 <xsd:element name="Base64Encoded" minOccurs="0"/>
44 <xsd:element name="ByReferenceURI" minOccurs="0"/>

```

```
45 <xsd:element name="Other" minOccurs="0">
46 <xsd:complexType>
47 <xsd:sequence>
48 <any minOccurs="0"
49     maxOccurs="unbounded"
50     processContents="lax"/>
51 </xsd:sequence>
52 </xsd:complexType>
53 </xsd:element>
54 </xsd:sequence>
55 </xsd:schema>
```


2 CRATE LANL DIDL SCHEMA: OAICRATE.XSD

```
1 <?xml version="1.0"?>
2 <xsd:annotation>
3 <xsd:documentation>
4     This is a schema to extend the oai_didl
5     content created by mod_oai with CRATE
6     model plugin information.
7     Schema author: Joan A. Smith,
8         Old Dominion University
9 </xsd:documentation>
10 </xsd:annotation>
11 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
12     xmlns="http://www.modoai.org/xmlns/"
13     elementFormDefault="qualified">
14 <xsd:element name="CrateUtilities"
15     minOccurs="1" maxOccurs="unbounded">
16 <xsd:complexType>
17 <xsd:element name="Plugin" type="PluginType"
18     minOccurs="1" maxOccurs="unbounded"/>
19 </xsd:complexType>
20 </xsd:element>
21 <xsd:complexType name="PluginType">
22 <xsd:sequence>
23 <xsd:element name="Name" type="xsd:string"
24     minOccurs="1" maxOccurs="1"/>
25 <xsd:element name="Exec" type="xsd:string"
26     minOccurs="0" maxOccurs="1"/>
27 <xsd:element name="Version" type="xsd:string"
28     minOccurs="1" maxOccurs="1"/>
29 <xsd:element name="MimeSet" type="xsd:string"
30     minOccurs="0" maxOccurs="1"/>
31 <xsd:any namespace="http://www.modoai.org/xmlns/"
32     minOccurs="0" maxOccurs="unbounded"/>
33 </xsd:sequence>
34 </xsd:complexType>
35 </xsd:schema>
```

APPENDIX B

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:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
5      http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
6  <responseDate>2008-05-15T00:04:08Z</responseDate>
7  <request verb="Identify">
8      http://www.foo.edu:8080/modoai</request>
9  <Identify>
10 <repositoryName>http://www.foo.edu:8080</repositoryName>
11 <baseURL>http://www.foo.edu:8080/modoai</baseURL>
12 <protocolVersion>2.0</protocolVersion>
13 <adminEmail>jsmit@cs.odu.edu</adminEmail>
14 <earliestDatestamp>1900-01-01T12:00:00Z</earliestDatestamp>
15 <deletedRecord>no</deletedRecord>
16 <granularity>YYYY-MM-DDThh:mm:ssZ</granularity>
17 <description>
18 <friends
19     xmlns="http://www.openarchives.org/OAI/2.0/friends/"
20     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
21     xsi:schemaLocation=
22         "http://www.openarchives.org/OAI/2.0/friends/
23         http://www.openarchives.org/OAI/2.0/friends.xsd">
24 </friends>
25 </description>
26 <description>
27 <gateway
28     xmlns="http://www.openarchives.org/OAI/2.0/gateway/"
29     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
30     xsi:schemaLocation=

```

```

31     "http://www.openarchives.org/OAI/2.0/gateway/
32     http://www.openarchives.org/OAI/2.0/gateway.xsd">
33 <source>http://www.foo.edu:8080</source>
34 <gatewayDescription>
35     http://www.modoai.org/gateway.html
36 </gatewayDescription>
37 <gatewayAdmin>mail@joanasmith.com</gatewayAdmin>
38 </gateway>
39 </description>
40 <description>
41 <modoai
42     xmlns:modoai="http://www.modoai.org/OAI/2.0/modoai/"
43     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
44     xsi:schemaLocation=
45         "http://www.modoai.org/OAI/2.0/modoai/modoai.xsd">
46 <modoai:version>0.7.1</modoai:version>
47 <modoai:server>
48     Apache/2.2.4 (Ubuntu)
49     PHP/5.2.3-1ubuntu6.3
50 </modoai:server>
51 </modoai>
52 </description>
53 <description>
54 <cratePlugins
55     xmlns="http://cratemodel.org/OAI/2.0/cratePlugins/"
56     xmlns:crateplugin="http://www.modoai.org/OAI/2.0/modoai/"
57     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
58     xsi:schemaLocation=
59         "http://cratemodel.org/OAI/2.0/cratePlugins/
60         http://cratemodel.org/OAI/2.0/cratePlugins.xsd">
61 <crateplugin>
62 <crateplugin:name>wc</crateplugin:name>
63 <crateplugin:exec>/usr/bin/wc</crateplugin:exec>
64 <crateplugin:version>
65 <![CDATA[wc (GNU coreutils) 5.97
66     Copyright (C) 2006 Free Software Foundation, Inc.
67     This is free software.
68     You may redistribute copies of it under the terms of
69     the GNU General Public License
70     <http://www.gnu.org/licenses/gpl.html>.
71     There is NO WARRANTY, to the extent permitted by law.
72     Written by Paul Rubin and David MacKenzie.]]>
73 </crateplugin:version>
74 <crateplugin:mimeType>text/*</crateplugin:mimeType>
75 </crateplugin>
76 <crateplugin>
77 <crateplugin:name>file</crateplugin:name>
78 <crateplugin:exec>/usr/bin/file</crateplugin:exec>
79 <crateplugin:version>
80 <![CDATA[file-4.21 magic file from
81     /etc/magic:/usr/share/file/magic]]>
82 </crateplugin:version>
83 <crateplugin:mimeType>*/*</crateplugin:mimeType>
84 </crateplugin>

```

```
85 <crateplugin>
86 <crateplugin:name>md5sum</crateplugin:name>
87 <crateplugin:exec>/usr/bin/md5sum</crateplugin:exec>
88 <crateplugin:version>
89 <![CDATA[md5sum (GNU coreutils) 5.97
90 Copyright (C) 2006 Free Software Foundation, Inc.
91 This is free software.
92 You may redistribute copies of it under the terms of
93 the GNU General Public License
94 <http://www.gnu.org/licenses/gpl.html>.
95 There is NO WARRANTY, to the extent permitted by law.
96 Written by Ulrich Drepper, Scott Miller,
97 and David Madore.]]>
98 </crateplugin:version>
99 <crateplugin:mimeType>application/*</crateplugin:mimeType>
100 </crateplugin>
101 <crateplugin>
102 <crateplugin:name>shasum</crateplugin:name>
103 <crateplugin:exec>/usr/bin/shasum</crateplugin:exec>
104 <crateplugin:version>
105 <![CDATA[shasum (GNU coreutils) 5.97
106 Copyright (C) 2006 Free Software Foundation, Inc.
107 This is free software.
108 You may redistribute copies of it under the terms of
109 the GNU General Public License
110 <http://www.gnu.org/licenses/gpl.html>.
111 There is NO WARRANTY, to the extent permitted by law.
112 Written by Ulrich Drepper, Scott Miller,
113 and David Madore.]]>
114 </crateplugin:version>
115 <crateplugin:mimeType>image/*</crateplugin:mimeType>
116 </crateplugin>
117 <crateplugin>
118 <crateplugin:name>sha224sum</crateplugin:name>
119 <crateplugin:exec>/usr/bin/sha224sum</crateplugin:exec>
120 <crateplugin:version>
121 <![CDATA[sha224sum (GNU coreutils) 5.97
122 Copyright (C) 2006 Free Software Foundation, Inc.
123 This is free software.
124 You may redistribute copies of it under the terms of
125 the GNU General Public License
126 <http://www.gnu.org/licenses/gpl.html>.
127 There is NO WARRANTY, to the extent permitted by law.
128 Written by Ulrich Drepper, Scott Miller,
129 and David Madore.]]>
130 </crateplugin:version>
131 <crateplugin:mimeType>image/png</crateplugin:mimeType>
132 </crateplugin>
133 <crateplugin>
134 <crateplugin:name>sha384sum</crateplugin:name>
135 <crateplugin:exec>/usr/bin/sha384sum</crateplugin:exec>
136 <crateplugin:version>
137 <![CDATA[sha384sum (GNU coreutils) 5.97
138 Copyright (C) 2006 Free Software Foundation, Inc.
```

```
139     This is free software.
140     You may redistribute copies of it under the terms of
141     the GNU General Public License
142     <http://www.gnu.org/licenses/gpl.html>.
143     There is NO WARRANTY, to the extent permitted by law.
144     Written by Ulrich Drepper, Scott Miller,
145     and David Madore.]]>
146 </crateplugin:version>
147 <crateplugin:mimeType>image/jpeg</crateplugin:mimeType>
148 </crateplugin>
149 <crateplugin>
150 <crateplugin:name>sha256sum</crateplugin:name>
151 <crateplugin:exec>/usr/bin/sha256sum</crateplugin:exec>
152 <crateplugin:version>
153 <![CDATA[sha256sum (GNU coreutils) 5.97
154     Copyright (C) 2006 Free Software Foundation, Inc.
155     This is free software.
156     You may redistribute copies of it under the terms of
157     the GNU General Public License
158     <http://www.gnu.org/licenses/gpl.html>.
159     There is NO WARRANTY, to the extent permitted by law.
160     Written by Ulrich Drepper, Scott Miller,
161     and David Madore.]]>
162 </crateplugin:version>
163 <crateplugin:mimeType>application/pdf</crateplugin:mimeType>
164 </crateplugin>
165 <crateplugin>
166 <crateplugin:name>zipinfo</crateplugin:name>
167 <crateplugin:exec>/usr/bin/zipinfo</crateplugin:exec>
168 <crateplugin:version>
169 <![CDATA[ZipInfo 2.42 of 28 February 2005,
170     by Greg Roelofs and the Info-ZIP group.]]>
171 </crateplugin:version>
172 <crateplugin:mimeType>*/zip</crateplugin:mimeType>
173 </crateplugin>
174 </cratePlugins>
175 </description>
176 </Identify>
177 </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 B– 1 on page 156 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:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4 xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
5 http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
6 <responseDate>2008-05-15T15:32:50Z</responseDate>
7 <request verb="GetRecord"
8 identifier="http://www.foo.edu/modoaitest/crate.jpeg"
9 metadataPrefix="oai_crate">http://www.foo.edu:8080/modoai/</request>
10 <GetRecord>
11 <record>
12 <header>
13 <identifier>http://www.foo.edu/modoaitest/crate.jpeg</identifier>
14 <datestamp>2000-02-28T17:00:00Z</datestamp>
15 <setSpec>mime:image/jpeg</setSpec>
16 </header>
17 <metadata>
18 <didl:DIDL xmlns:didl="urn:mpeg:mpeg21:2002:02-DIDL-NS"
19 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
20 xsi:schemaLocation="urn:mpeg:mpeg21:2002:02-DIDL-NS
21 http://purl.lanl.gov/STB-RL/schemas/2004-11/DIDL.xsd">
22 <didl:Item>
23 <didl:Descriptor>
24 <didl:Statement mimeType="application/xml; charset=utf-8">
25 <dii:Identifier xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
26 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
27 xsi:schemaLocation="urn:mpeg:mpeg21:2002:01-DII-NS
28 http://purl.lanl.gov/STB-RL/schemas/2003-09/DII.xsd">
29 http://www.foo.edu/modoaitest/crate.jpeg</dii:Identifier>
30 </didl:Statement>
31 </didl:Descriptor>
32 <didl:Descriptor>
33 <didl:Statement mimeType="application/xml; charset=utf-8">
34 <http:header xmlns:http="http://www.modoai.org/OAI/2.0/http_header/"
35 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
36 xsi:schemaLocation="http://www.modoai.org/OAI/2.0/http_header/
37 http://purl.lanl.gov/STB-RL/schemas/2004-08/HTTP-HEADER.xsd">
38 <http:Content-Length>19339</http:Content-Length>
39 <http:Server>Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6.3</http:Server>
40 <http:Content-Type>image/jpeg</http:Content-Type>
41 <http>Last-Modified>Mon, 28 Feb 2000 17:00:00 GMT</http>Last-Modified>
```

```

42 <http:Date>Thu, 15 May 2008 15:32:50 GMT</http:Date>
43 <http:Via>1.1 SRVWINISA003</http:Via><http:User-Agent>Mozilla/4.0
44 (compatible; MSIE 5.5; Windows 98)</http:User-Agent>
45 <http:Host>www.foo.edu:8080</http:Host>
46 <http:Te>deflate,gzip;q=0.3</http:Te>
47 <http:X-Forwarded-For>70.161.101.174</http:X-Forwarded-For>
48 <http:Accept>image/gif, image/x-xbitmap,
49 image/jpeg, image/pjpeg, application/vnd.ms-excel,
50 application/msword, application/vnd.ms-powerpoint,
51 */*</http:Accept>
52 <http:Connection>Keep-Alive</http:Connection></http:header>
53 </didl:Statement>
54 </didl:Descriptor>
55 <oai_crate:crateplugin
56 xmlns:oai_crate='http://modoai.org/OAI/2.0/oai_crate/'
57 xmlns:crateplugin="http://modoai.org/OAI/2.0/crateplugin "
58 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance "
59 xsi:schemaLocation="http://modoai.org/OAI/2.0/oai_crate/crate.xsi" >
60 <crateplugin>
61 <crateplugin:name>file</crateplugin:name>
62 <crateplugin:version>
63 <![CDATA[file-4.21 magic file from /etc/magic:/usr/share/file/magic
64 ]]>
65 </crateplugin:version>
66 <crateplugin:content>
67 <![CDATA[
68 /var/www/modoaitest/crate.jpeg:
69 JPEG image data, JFIF standard 1.01
70 ]]>
71 </crateplugin:content>
72 </crateplugin>
73 <crateplugin>
74 <crateplugin:name>shalsum</crateplugin:name>
75 <crateplugin:version>
76 <![CDATA[shalsum (GNU coreutils) 5.97
77 Copyright (C) 2006 Free Software Foundation, Inc.
78 This is free software. You may redistribute copies of it under the
79 terms of the GNU General Public License
80 <http://www.gnu.org/licenses/gpl.html>.
81 There is NO WARRANTY, to the extent permitted by law.
82 Written by Ulrich Drepper, Scott Miller, and David Madore.
83 ]]>
84 </crateplugin:version>
85 <crateplugin:content>
86 <![CDATA[7b15663fbfb3bc174c5883d2b57facbe91465bcb
87 /var/www/modoaitest/crate.jpeg
88 ]]>
89 </crateplugin:content>
90 </crateplugin>
91 <crateplugin>
92 <crateplugin:name>sha384sum</crateplugin:name>
93 <crateplugin:version>
94 <![CDATA[sha384sum (GNU coreutils) 5.97
95 Copyright (C) 2006 Free Software Foundation, Inc.

```

```
96 This is free software. You may redistribute copies of it under the
97 terms of the GNU General Public License
98 <http://www.gnu.org/licenses/gpl.html>.
99 There is NO WARRANTY, to the extent permitted by law.
100 Written by Ulrich Drepper, Scott Miller, and David Madore.
101 ]]>
102 </crateplugin:version>
103 <crateplugin:content>
104 <![CDATA[709f27aa6832e044fb7edaab2abd0fec00cb478dddeb68674
105 4369c6bbd1bbb1aa2052ecdf58a73b1945dd69150583bb6
106 /var/www/modoaitest/crate.jpeg
107 ]]>
108 </crateplugin:content>
109 </crateplugin>
110 </oai_crate:crateplugin>
111 <didl:Component>
112 <didl:Resource mimeType="image/jpeg"
113 encoding="base64">/9j/4AAQSkZJRgBDAAE.....IWzr9e3+1P/2Q==
114 </didl:Resource>
115 <didl:Resource mimeType="image/jpeg"
116 ref="http://www.foo.edu/modoaitest/crate.jpeg"/>
117 </didl:Component>
118 </didl:Item>
119 </didl:DIDL>
120 </metadata>
121 </record>
122 </GetRecord>
123 </OAI-PMH>
```


3 OAI-PMH LIST IDENTIFIERS RESPONSE

The presence of CRATE plugins in MODDOI 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 E on page 183 for more details on sitemap files.

REQUEST:

```
http://www.foo.edu:8080/modoai?verb=ListIdentifiers
```

RESPONSE:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
5  http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
6  <responseDate>2008-05-15T15:31:10Z</responseDate>
7  <request verb="ListIdentifiers" metadataPrefix="oai_crate">
8  http://www.foo.edu:8080/modoai/</request>
9  <ListIdentifiers>
10 <header>
11 <identifier>http://www.foo.edu:8080/modoaitest/NetBeansAntTasks.zip
12 </identifier>
13 <datestamp>2007-09-19T08:06:56Z</datestamp>
14 <setSpec>mime:application:zip</setSpec>
15 </header>
16 <header>
17 <identifier>http://www.foo.edu:8080/modoaitest/README.txt</identifier>
18 <datestamp>2008-04-26T16:37:57Z</datestamp>
19 <setSpec>mime:text:plain</setSpec>
20 </header>
21 <header>
22 <identifier>http://www.foo.edu:8080/modoaitest/crate.jpeg</identifier>
23 <datestamp>2000-02-28T17:00:00Z</datestamp>
24 <setSpec>mime:image:jpeg</setSpec>
25 </header>
26 <header>
27 <identifier>http://www.foo.edu:8080/modoaitest/diag.jpg</identifier>
28 <datestamp>2002-10-22T16:00:00Z</datestamp>
29 <setSpec>mime:image:jpeg</setSpec>
30 </header>
31 <header>
32 <identifier>http://www.foo.edu:8080/modoaitest/file.sxw</identifier>
33 <datestamp>2005-12-31T17:00:00Z</datestamp>
34 <setSpec>mime:application:vnd.sun.xml.writer</setSpec>
35 </header>
36 <resumptionToken>5*oai_crate*0*0*0</resumptionToken></ListIdentifiers>
37 </OAI-PMH>

```

APPENDIX C

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 61 on the next page. 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. 61: .

“Two foos having lunch” (foo2.jpg). Appendix C–1 shows the metadata provided from the HTTP response, the Linux file utility, Pronom-DROID, Exif Tool, 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 Exif Tool

Generally, Exif Tool was written to analyze digital camera photographs. While our sample Figure 61 on the preceding page 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
```

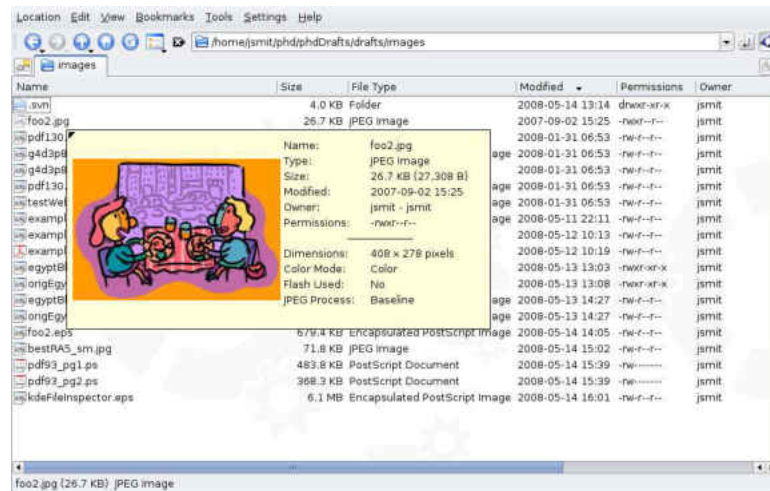


FIG. 62: .

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 62.

```
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. 63: 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 on 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 61 on page 165 as shown in Appendix C on page 164. The following analysis is produced by Exif Tool applied to the photograph shown in Figure 63.

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] @ 210
EXIF SubIFD at offset 418:
ExposureTime: (0x829A) Ratio=10/2500 @ 832
FNumber: (0x829D) Ratio=33/10 @ 840
ExposureProgram: (0x8822) Short=Program Normal @ 452
ISO Speed Ratings: (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

```

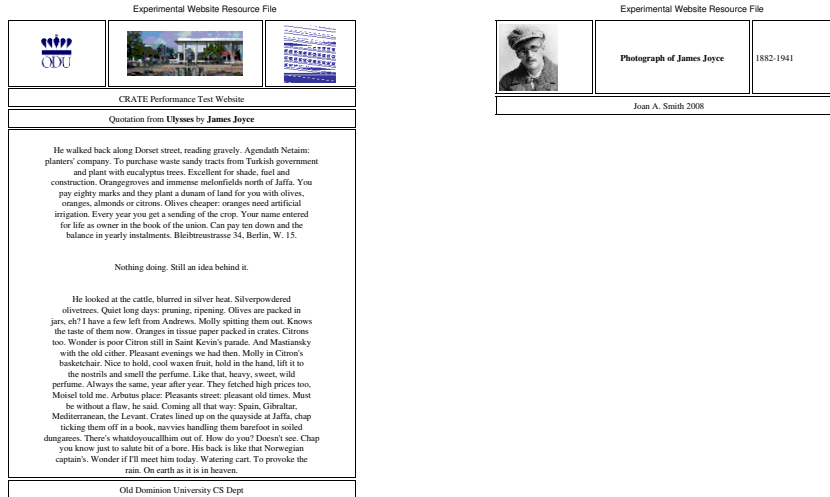



FIG. 64: Sample PDF analyzed by Jhove

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 64 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. When used to inspect the same PDF shown in Figure 64 on page 171, it produces the following output.

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>Adobe PDF</Format> <Version>1.3</Version>
</FileFormat>
<Text>
<CharacterSet>ISO-8859-1</CharacterSet>
<MarkupLanguage>unknown</MarkupLanguage>
</Text></File></Files>
</Object>
```



FIG. 65: 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 65.

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 [130].

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 [130], 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 web sites 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 web sites, anyone involved with digital libraries can easily point out the many differences between everyday web sites and a true Digital Library (DL). The Web is an unorganized amalgamation of digital pages with little metadata and unpredictable additions, deletions, and modifications — a crawlalooza for the web robot. For an archiving repository seeking to preserve web sites, the site preparation process is challenging thanks to the wide variety of resource types and content that exist on web sites. Typically, an archivist will crawl the target web site 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 web site 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 D

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 D– 2 on page 181. 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 29 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 be named. Thus, lines 11 – 24 create a single variable called `modoai_plugin` with 11 instances of

TABLE 29: 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

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 interpreted as shown in Table 30. Appendix D– 2 on page 181 contains the actual configuration directives used for the CRATE utilities experiments discussed in Chapter VIII on page 93.

TABLE 30: MODOAI plugin elements & attributes

<i>modoai_plugin</i>	<i>wc</i>	<i>/usr/bin/wc %s</i>	<i>/usr/bin/wc -v</i>	<i>text/*</i>
add a plugin to the list	name is “wc”	executable command %s: filename	command to print version info	use on any text file
<i>modoai_plugin</i>	<i>exifTool</i>	<i>/usr/bin/exiftool -a -u %s</i>	<i>/usr/bin/exiftool -v</i>	<i>image/*</i>
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 D– 3 on page 182. 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'
14      '/usr/bin/md5sum --version' application/*
15  modoai_plugin shasum '/usr/bin/shasum %s'
16      '/usr/bin/shasum --version' image/*
17  modoai_plugin sha224sum '/usr/bin/sha224sum %s'
18      '/usr/bin/sha224sum --version' image/png
19  modoai_plugin sha384sum '/usr/bin/sha384sum %s'
20      '/usr/bin/sha384sum --version' image/jpeg
21  modoai_plugin sha256sum '/usr/bin/sha256sum %s'
22      '/usr/bin/sha256sum --version' application/pdf
23  modoai_plugin zipinfo '/usr/bin/zipinfo %s'
24      '/usr/bin/zipinfo --version' */zip
25  modoai_plugin jhove "/opt/jhove/jhove -c /opt/jhove/conf/jhove.conf
26      -m jpeg-hul -h xml %s" "/opt/jhove/jhove
27      -c /opt/jhove/conf/jhove.conf -h xml -v" image/jpeg
28  modoai_plugin pronom_droid "/opt/jdk1.5.0_07/bin/java -jar
29      /opt/droid/DROID.jar
30      -L%s -S/opt/droid/DROID_SignatureFile_V12.xml"
31      "/opt/jdk1.5.0_07/bin/java -jar /opt/droid/DROID.jar -V" */*
32  modoai_plugin exifTool "/usr/bin/exiftool -a -u %s"
33      "/usr/bin/exiftool -ver" image/*
34  modoai_plugin ots '/usr/local/bin/ots -a %s'
35      '/usr/local/bin/ots -v' text/*
36  modoai_plugin metadata-extractor
37      "/home/jsmit/metadata-extractor/dist/extract.sh extract
38      'NLNZ Data Dictionary' Default simple 'Blah' 3333 %s
39      2>/dev/null %s"
40      "/bin/echo metadata-extractor 3.4" image/jpeg
41  modoai_plugin dcdot '/opt/dublinCoreUtil/dcdot/jasDC.pl %s'
42      '/opt/dublinCoreUtil/dcdot/jasDC.pl --version' text/html
43 </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 E

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 [5, 47] 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. [4, 150] These accept a starting URL and proceed to follow all of the links found (Figure 66 on the following page) 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. [126] 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>
  <lastmod>2007-11-21T14:35:21Z</lastmod>
</url>
```

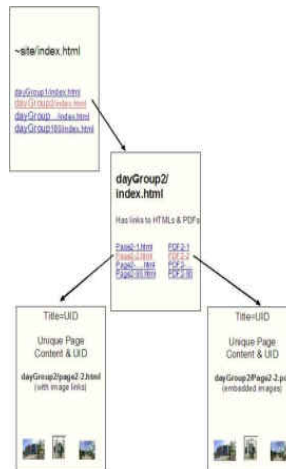


FIG. 66: Links on a web page provide crawlers with a list of resources to ask for. Usually only *internal* links are followed.

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/softdrink_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

2004–Present 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 SOCIETES

Association for Computing Machinery (ACM)

Institute of Electrical and Electronics Engineers (IEEE)

Alpha Gamma Sigma National Honor Society (ΑΓΣ)

Golden Key International Honor Society

PERMALINKS

Home Page: <http://www.joanasmith.com/>

Email: mail@joanasmith.com

Typeset using L^AT_EX.