SIMPLE DIGITAL LIBRARIES

LIGHTON PHIRI

A DISSERTATION SUBMITTED
IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN
COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCE
UNIVERSITY OF CAPE TOWN

SUPERVISED BY
HUSSEIN SULEMAN

DECEMBER 2013

Simple Digital Libraries

by

Lighton Phiri

Plagiarism Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for which is properly acknowledged, is my own.

Lighton Phiri

Friday August 10, 2013

(Date)

****

Acknowledgements

First of all, I would like to thank my supervisor, Professor Hussein Suleman, for giving me the opportunity to work with him; and for his encouragement, technical advice and support throughout my graduate studies. He read carefully early drafts of the manuscript, and his suggestions and proposed corrections contributed to the final form of this thesis. I am very grateful for this.

In addition, a number of individuals implicitly and explicitly contributed to this thesis in one way or the other. To these people, I would like to express my sincere thanks. In particular, I would like to thank Nicholas Wiltshire for facilitating access to the Spatial Archaeology Research Unit (SARU) Rock Art digital collection; Miles Robinson and Stuart Hammar for basing their "Bonolo" honours project on the Bleek and Lloyd case study collection; Kaitlyn Crawford, Marco Lawrence and Joanne Marston for basing their "School of Rock Art" honours project on the SARU Rock Art archaeological database case study collection; the University of Cape Town Computer Science Honours class of 2012 for taking part in the developer survey; and especially Kyle Williams for his willingness to help.

Furthermore, I would like to thank the Centre for Curating the Archive at the University of Cape Town and the Department of Archeology at the University of Cape Town for making available the digital collections that were used as case studies. I would also like to thank the Networked Digital Library of Thesis and Dissertation (NDLTD) for implicitly facilitating access to the dataset used in performance experiments through their support for open access to scholarship.

Finally, I would like to express my sincere gratitude to my family for their support during my long stay away from home.

To my parents, and my 'banded' brothers.

Abstract

The design of Digital Library Systems (DLSes) has evolved overtime, both in sophistication and complexity, to complement the complex nature and sheer size of digital content being curated. However, there is also a growing demand from content curators, with relatively small-size collections, for simpler and more manageable tools and services to manage their content. The reasons for this particular need are driven by the assumption that simplicity and manageability might ultimately translate to lower costs of maintenance of such systems.

This research proposes and advocates for a minimalist and simplistic approach to the overall design of DLSes. It is hypothesised that Digital Library (DL) tools and services based on such designs could potentially be easy to use and manage.

A meta-analysis of existing DL and non-DL tools was conducted to aid the derivation of design principles for simple DLSes. The design principles were then mapped to design decisions applied to the design of a prototype simple repository. In order to assess the effectiveness of the simple repository design, two real-world case study collections were implemented based on the design. In addition, a developer-oriented study was conducted using one of the case study collections to evaluate the simplicity and ease of use of the prototype system. Furthermore, performance experiments were conducted to establish the extent to which such a simple design approach would scale and also establish comparative advantages to existing designs.

In general, the study outlined some possible implications of simplifying DLS design; specifically the results from the developer-oriented user study indicate that simplicity in the design of the DLS repository sub-layer does not severely impact the interaction between the service sub-layer and the repository sub-layer. Furthermore, the scalability experiments indicate that desirable performance results for small- and medium-sized collections are attainable.

The practical implication of the proposed design approach is two-fold: firstly the minimalistic design has the potential to be used to design simple and yet easy to use tools with comparable features to those exhibited by well-established DL tools; and secondly, the principled design approach has the potential to be applied to the design of non-DL application domains.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

5S	Streams, Structures, Spaces and Societies.
AHP	Analytic Hierarchy Process.
DL	Digital Library.
DLMS	Digital Library Management System.
DLS	Digital Library System.
DOI	Digital Object Identifier.
ETD	Electronic Thesis and Dissertation.
FLOSS	Free/Libre/Open Source Software.
NDLTD	Networked Digital Library of Thesis and Dissertation.
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting.
OASIS	Organisation for the Advancement of Structured Information Standards.
PDF/A	PDF/A is an ISO-standardised version of the Portable Document Format (PDF).
PURL	Persistent Uniform Resource Locator.
RAP	Repository Access Protocol.
SARU	Spatial Archaeology Research Unit.
URI	Uniform Resource Identifier.
WWW	World Wide Web Technologies.
XML	Extensible Markup Language.

Chapter 1

Introduction

The last few decades has seen an overwhelming increase in the amount of digitised and born digital information. There has also been a growing need for specialised systems tailored to better handle this digital content. Digital Libraries (DLs) are specifically designed to store, manage and preserve digital objects over long periods of time. Figure 1-1 illustrates a high-level view of a typical Digital Library System (DLS) architecture.

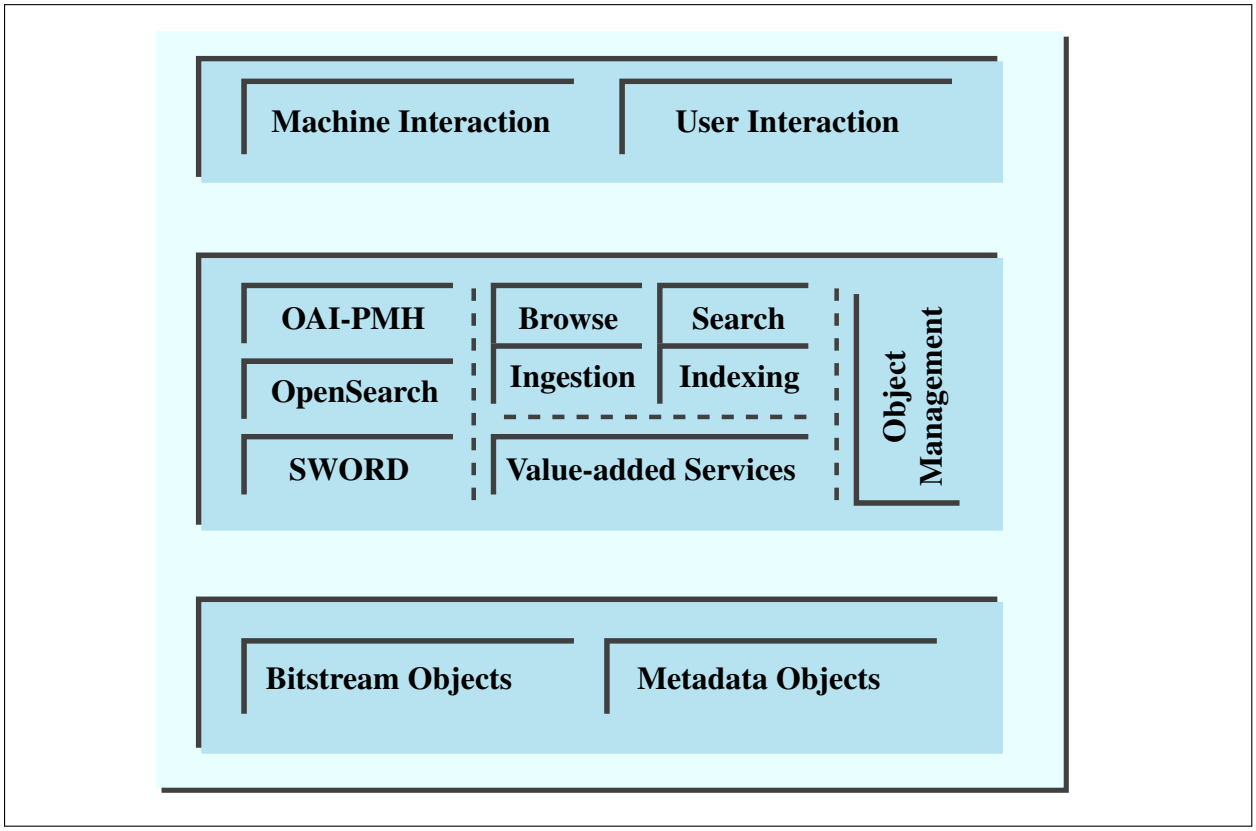


Figure 1-1. High level architecture of a typical Digital Library System

1.1 Motivation

DLs began as an abstraction layered over databases to provide higher level services (Arms, Blanchi, and Overly, 1997; Baldonado et al., 1997; Frew et al., 1998) and have evolved, subsequently making them complex (Janée and Frew, 2002; Lagoze et al., 2006) and difficult to maintain, extend and reuse. The difficulties resulting from the complexities of such tools are especially prominent in organisations and institutions that have limited resources to manage such tools and services. Some examples of organisations that fall within this category include cultural heritage organisations and a significant number of other organisations in developing countries found in regions such as Africa (Suleman, 2008).

The majority of existing platforms are arguably unsuitable for resource-constrained environments due to the following reasons:

- Some organisations do not have sustainable funding models, making it difficult to effectively manage the preservation life-cycle as most tools are composed of custom and third-party components that require regular updates.
- A number of existing tools require technically-inclined experts to manage them, effectively raising their management costs.
- The majority of modern platforms are bandwidth intensive. However, they sometimes end up being deployed in regions were Internet bandwidth is unreliable and mostly very expensive, making it difficult to guarantee widespread accessibility to services offered.

A potential solution to this problem is to explicitly simplify the overall design of DLSes so that the resulting tools and services are more easily adopted and managed over time. This premise is drawn from the many successes of the application of minimalism, as discussed in Section 2.5. In light of that, this research proposes the design of lightweight tools and services, with the potential to be easily adopted and managed.

1.2 Hypotheses

This research was guided by three working hypotheses that are a direct result of grounding work previously conducted (Suleman, 2007; Suleman et al., 2010). The three hypotheses are as follows:

- A formal simplistic abstract framework for DLS design can be derived.
- A DLS architectural design based on a simple and minimalistic approach could be potentially easy to adopt and manage over time.
- The system performance of tools and services based on simple architectures could be adversely affected.

1.3 Research questions

The core of this research was aimed at investigating the feasibility of implementing a DLS based on simplified architectural designs. In particular, the research was guided by the following research questions:

Is it feasible to implement a DLS based on simple architectures?

This primary research question was broadly aimed at investigating the viability of simple architectures. To this end, the following secondary questions were formulated to clarify the research problem.

i **How should simplicity for DLS storage and service architectures be defined?**

This research question served as a starting point for the research, and was devised to help provide scope and boundaries of simplicity for DLS design.

ii **What are the potential implications of simplifying DLS—adverse or otherwise?**

It was envisaged, from the onset, that simplifying the overall design of a DLS would potentially result in both desirable and undesirable outcomes. This research question was thus aimed at identifying the implications of simplifying DLS design.

iii **What are some of the comparative advantages and disadvantages of simpler architectures to complex ones?**

A number of DLS architectures have been proposed over the past two decades, ranging from those specifically designed to handle complex objects to those with an overall goal of creating and distributing collection archives (see Section 2.4). This research question was aimed at identifying some of the advantages and disadvantages of simpler architectures compared to well-established DL architectures. This includes establishing how well simple architectures support the scalability collections.

1.4 Scope and approach

Table 1-1 shows a summary of the research process followed to answer the research questions.

Table 1-1. Summary of research approach process

Research Process	Procedure
Literature synthesis	Preliminary review of existing literature
Research proposal	Scoping and formulation of research problem
Exploratory study	Derivation of design principles
Repository design	Mapping of design principles to design process
Case studies	Implementation case study collections
Evaluation	Experimentation results and discussion

1.5 Thesis outline

This manuscript is structured as follows:

- Chapter 1 serves as an introduction, outlining the motivation, research questions and scope of the research conducted.
- Chapter 2 provides background information and related work relevant to the research conducted.
- In Chapter 3 the exploratory study that was systematically conducted to derive a set of design principles is described, including the details of the principles derived.
- Chapter 4 presents a prototype repository whose design decisions are directly mapped to some design principles outlined in Chapter 3.
- Chapter 5 describes two real-world case study implementation designed and implemented using the repository design outlined in Chapter 4.
- The implications of the prototype repository design are outlined in Chapter 6 through: experimental results from a developer-oriented survey conducted to evaluate the simplicity and extensibility; and through scalability performance benchmark results of some DLS operations conducted on datasets of different sizes.
- Chapter 7 highlights concluding remarks and recommendations for potential future work.

Chapter 2

Background

Research in the field of DLs has been going on for over two decades. The mid 1990s, in particular, saw the emergence of a number of government funded projects (Griffin, 1998), conferences (Adam, Bhargava, and Yesha, 1995), technical committees (*Dublin Core Metadata Element Set, Version 1.1* 1999; Lorist and Meer, 2001) and workshops (Dempsey and Weibel, 1996; Lagoze, Lynch, and Daniel, 1996), specifically set up to foster formal research in the field of DLs. The rapid technological advances and, more specifically, Web technologies have resulted in a number of different DLS frameworks, conceptual models, architectural designs and DL software tools. The variation in the designs can largely be attributed to the different design goals and corresponding specific problems that the solutions were aimed to address.

This chapter is organised as follows. Section 2.1 presents an overview of DLs, including definitions and sample application domains; Section 2.2 introduces fundamental key concepts behind DLs; Section 2.3 is a discussion of pioneering work on some proposed frameworks and reference models that have been applied to the implementation of DLS; Section 2.4 presents related work through a discussion of some popular Free/Libre/Open Source Software (FLOSS) tools used for managing digital collections; Section 2.5 broadly discusses designs whose successes are hinged on simplicity; Section 2.6 discusses some commonly used storage solutions; and finally Section 2.7 presents two prominent methods used to capture software design decisions.

2.1 Digital Libraries

2.1.1 Definitions

The field of DLs is a multidisciplinary field that comprises disciplines such as data management, digital curation, document management, information management, information retrieval and library sciences. Fox et al. (Fox et al., 1995) outline the varying impressions of DLs from persons in different disciplines and adopt a pragmatic approach of embracing the different definitions. They further acknowledge the metaphor of the traditional library as empowering and recognise the importance of knowledge systems that have evolved as a result. Arms (see Arms, 2001, chap. 1)

provides an informal definition by viewing a DL indexDigital Libraries as a well organised, managed network-accessible collection of information—with associated services.

In an attempt to overcome the complex nature of DLs, Gonçalves et al. (Gonçalves et al., 2004) define a DL, using formal methods, by constructively defining a minimal set of components that make up a DL. The set-oriented and functional mathematical formal basis of their approach facilitates the precise definition of each component as functional compositions.

The European Union co-funded DELOS Network of Excellence on DLs working group proposed a reference model and drafted The DL Manifesto with the aim of setting the foundations and identifying concepts within the universe of DLs (Candela et al., 2007). The DELOS DL indexDigital Libraries reference model envisages a DL indexDigital Libraries universe as a complex framework and tool having no logical, conceptual, physical, temporal or personal borders or barriers on information. A DL indexDigital Libraries is perceived as an evolving organisation that comes into existence through a series of development steps that bring together all the necessary constituents, each corresponding to three different levels of conceptualisation of the universe of DLs (Candela et al., 2008). The DELOS DL indexDigital Libraries reference model is discussed in depth in Section 2.3.3.

2.1.2 Application domains

The use of DLs has become widespread mainly due to the significant technological advances that have been taking place since the 1990s. The advent of the Internet has particularly influenced this widespread use. There are various application domains in which DLs are used and researchers are continuously coming up with innovative ways of increasing the footprint of DL indexDigital Libraries usage.

Academic institutions are increasingly setting up institutional repositories to facilitate easy access to research output. DLs play a vital role by ensuring that intellectual output is collected, managed, preserved and later accessed efficiently and effectively. Figure 2-1 is an illustration of an institutional repository system—a full text open access institution repository of the Copperbelt University[1].

Cultural heritage organisations are increasingly digitising historical artifacts in a quest to display them online to a much wider audience. In light of this, DLSes are being developed to enable easy access to this information. Figure 2-2 is a screen snapshot of the Digital Bleek and Lloyd Collection[2], which is a digital collection of historical artifacts that document the culture and language of the |Xam and !Kun groups of Bushman people of Southern Africa.

There has also been an increasing number of large scale archival projects that have been initiated to preserve human knowledge and provide free access to vital information (Hart, 1992).

In addition, a number of federated services are increasingly being implemented with the aim of making information from heterogeneous services available in centralised location. Figure 2-3 shows a snapshot of the South African National Electronic Thesis and Dissertation (NETD) por-

[1] http://dspace.cbu.ac.zm:8080/jspui
[2] http://lloydbleekcollection.cs.uct.ac.za

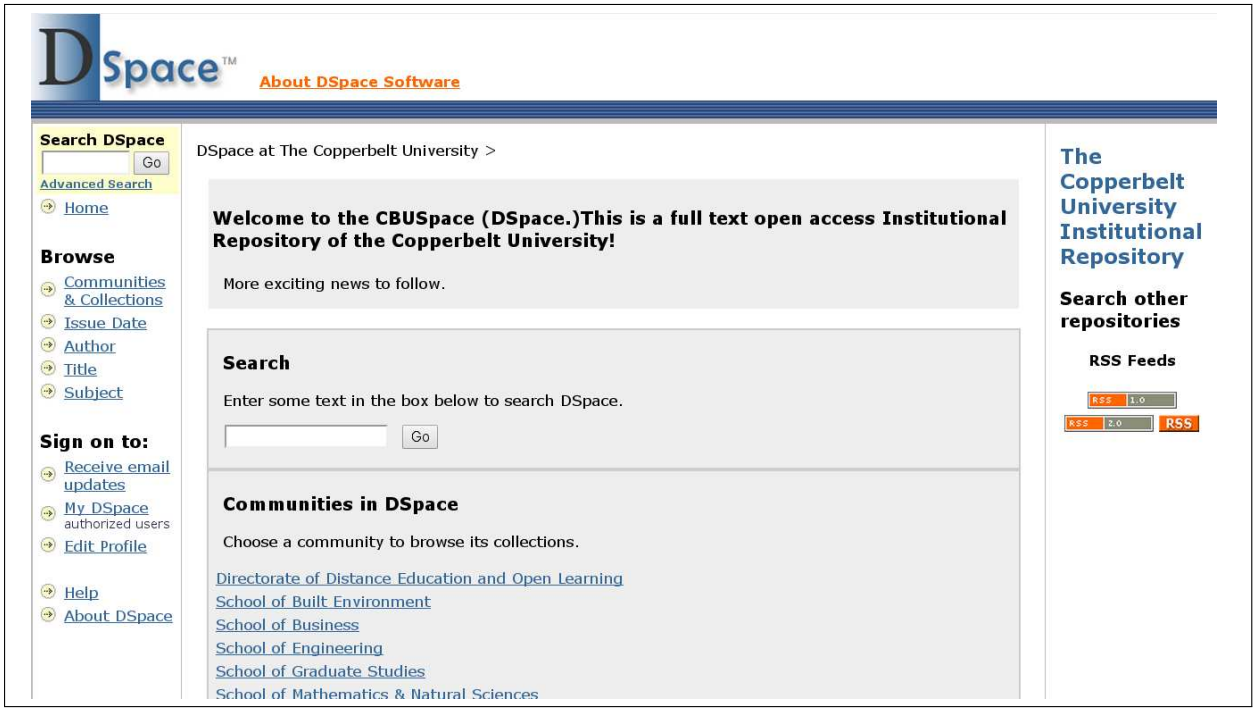


Figure 2-1. Screenshot showing the Copperbelt University institution repository

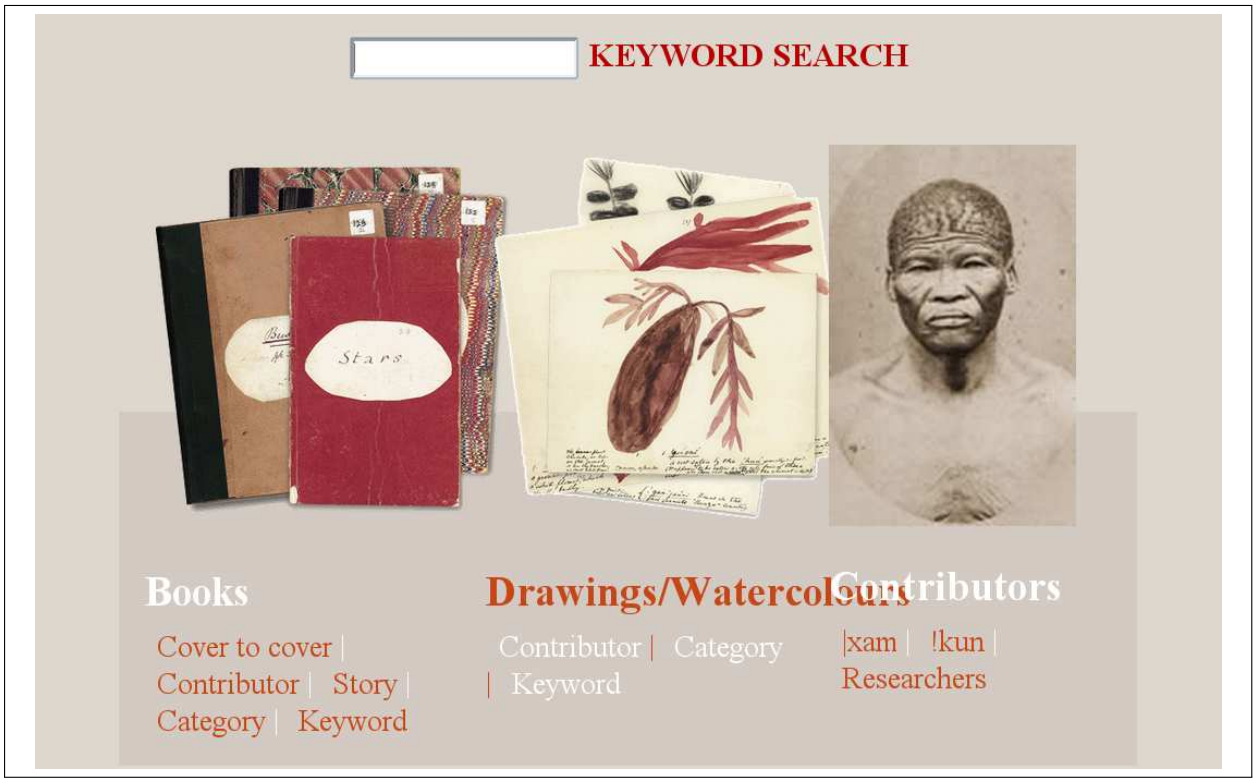


Figure 2-2. Screenshot showing the digital Bleek& Lloyd collection

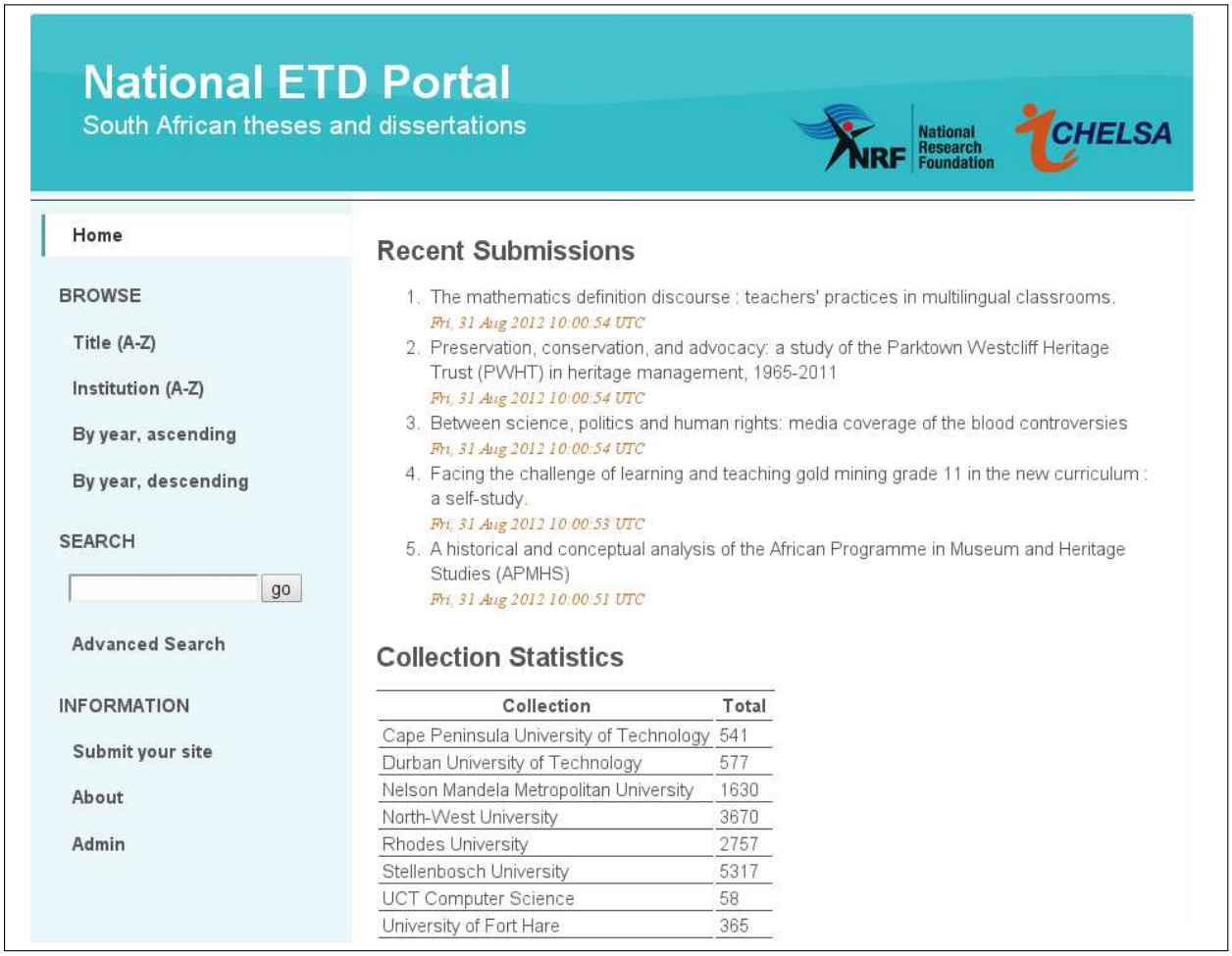


Figure 2-3. Screenshot showing the South African National Electronic Thesis and Dissertation portal

tal—a federated service that makes it possible for Electronic Thesis and Dissertations (ETDs) from various South African universities to be discovered from a central location.

2.1.3 Summary

The massive number of physical copies being digitised, coupled with the increase in the generation of born-digital objects, has created a need for tools and services—DLs—for making these objects easily accessible and preservable over long periods of time. The importance of these systems is manifested through their ubiquitous use in varying application domains.

This section broadly defined and described DLs, and subsequently discussed some prominent application domains within which are currently used.

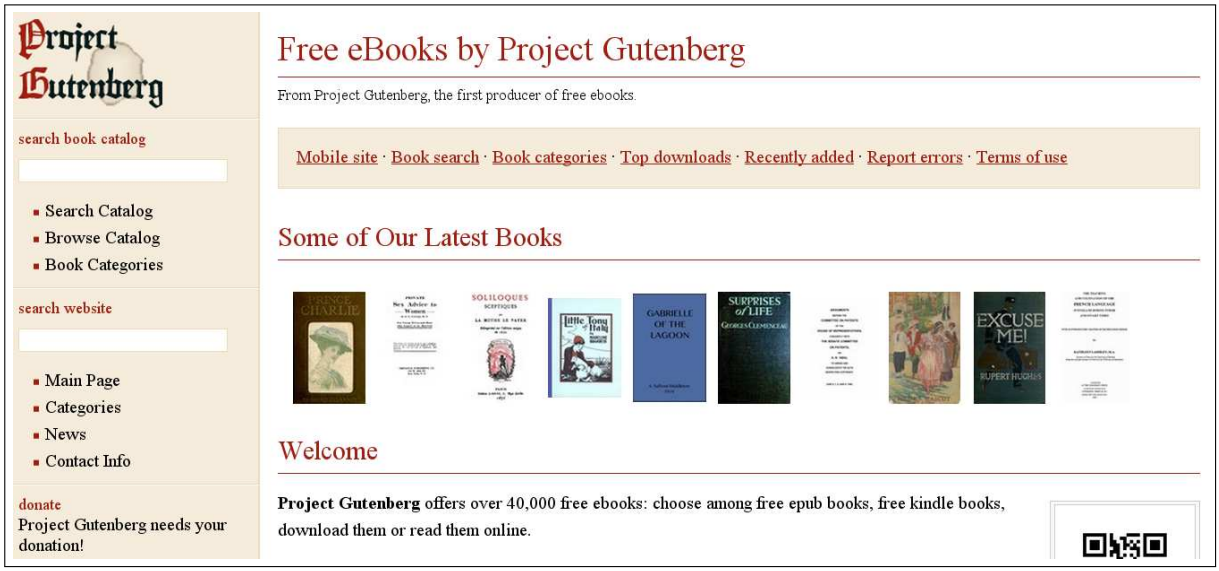


Figure 2-4. Screenshot showing the Project Gutenburg free ebooks portal

2.2 Fundamental concepts

2.2.1 Identifiers

An identifier is a name given to an entity for current and future reference. Arms (Arms, 1995) classifies identifiers as vital building blocks for DL and emphasises their role in ensuring that individual digital objects are easily identified and changes related to the objects are linked to the appropriate objects. He also notes that they are also essential for information retrieval and for providing links between objects.

The importance of identifiers is made evident by the widespread adoption of standardised naming schemes such as Digital Object Identifiers (DOIs)[3] (Paskin, 2005; Paskin, 2010) , Handles System[4] and Persistent Uniform Resource Locators (PURLs)[5].

Uniform Resource Identifiers (URIs) (Berners-Lee, Fielding, and Masinter, 2005) are considered a suitable naming scheme for digital objects primarily because they can potentially be resolved through standard Web protocols; that facilitates interoperability, a feature that is significant in DL whose overall goal is the widespread dissemination of information.

2.2.2 Interoperability

Interoperability is a system attribute that enables a system to communicate and exchange information with other heterogeneous systems in a seamless manner. Interoperability makes it possible for services, components and systems developed independently to potentially rely on one another

[3] http://www.doi.org
[4] http://www.handle.net
[5] http://purl.oclc.org

to accomplish certain tasks with the overall goal of having individual components evolve independently, but be able to call on each other, thus exchanging information, efficiently and conveniently (Paepcke et al., 1998). DL interoperability has particularly made it possible for federated services (Gonçalves, France, and Fox, 2001) to be developed, mainly due to the widespread use of the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH).

There are various protocols that have been developed to facilitate interoperability among heterogeneous DLSes. Prominent interoperability protocols include: Z39.50 (Lynch, 1991) a client-server protocol used for remote searching; OAI-PMH (Lagoze et al., 2002b), which has been extensively used for metadata harvesting; and RSS (Winer, 2007), a Web based feed format commonly used for obtaining updates on Web resources.

Extensible Markup Language (XML) has emerged as the underlying language used to support a number of these interoperability protocols, largely due to its simplicity and platform independence.

2.2.3 Metadata

Metadata is representational information that includes pertinent descriptive annotations necessary to understand a resource. Arms (Arms, Blanchi, and Overly, 1997) describes different categories of information as being organised as sets of digital objects—a fundamental unit of the DL architecture—that are composed of digital material and key-metadata. He defines the key-metadata as information needed to manage the digital object in a networked environment. The role performed by metadata is both implicit and explicit and its functions can be more broadly divided into distinct categories. A typical digital object normally has administrative metadata for managing the digital object, descriptive metadata to facilitate the discovery of information, structural metadata for describing relationships within the digital object and preservation metadata that stores provenance information. Metadata is made up of elements that are grouped into a standard set, to achieve a specific purpose, resulting in a metadata schema. There are a number of metadata schemes that have been developed as standards across various disciplines and they include, among others, Dublin Core (*Dublin Core Metadata Element Set, Version 1.1* 1999), Learning Object Metadata (LOM) (*Draft Standard for Learning Object Metadata* 2002), Metadata Encoding and Transmission Standard (METS)[6] and Metadata Object Description Schema (MODS)[7]. Metadata can either be embedded within the digital object—as is the case with Portable Document Format (PDF) and Hypertext Transfer Markup Language (HTML) documents—or stored separately with links to the resources being described. Metadata in DL is often stored in databases for easy management and access.

2.2.4 Standards

The fast pace at which technology is moving has spawned different types of application software tools. This means that the choice of which technology to use in any given instance differs, thus complicating the process of integrating application software with other heterogeneous software

[6] http://www.loc.gov/standards/mets
[7] http://www.loc.gov/standards/mods

tools. Standards become particularly useful in such situations because they form the basis for developing interoperable tools and services. A standard is a specification—a formal statement of a data format or protocol—that is maintained and endorsed by a recognised standards body (see Suleman, 2010, chap. 2).

Adopting and adhering to standards has many other added benefits—and Strand et al. (Strand, Mehta, and Jairam, 1994) observe that applications that are built on standards are more readily scalable, interoperable and portable, constituting software quality attributes that are important for the design, implementation and maintenance of DLs. Standards also play a vital role in facilitating long term preservation of digital objects by ensuring that documents still become easily accessible in the future. This is done by ensuring that the standard itself does not change and by making the standard backwards compatible. Notable use of standards in DL include the use of XML as the underlying format for metadata and OAI-PMH as an interoperability protocol. Digital content is also stored in well known standards, as is the case with documents that are normally stored in PDF/A format. The use of standards in DLSes, however, has its own shortcomings; in certain instances, the use of standards can be a very expensive venture as it may involve a lot of cross-domain effort (Lorist and Meer, 2001).

2.2.5 Summary

A DLS operates as a specialised type of information system and exhibits certain characteristics to attain its objects. This section discussed fundamental concepts, associated to DLSes, that help form the necessary building blocks for implementing DLs.

2.3 Digital Libraries frameworks

A reference model is an abstract framework that provides basic concepts used to understand the relationships among items in an environment. The Organisation for the Advancement of Structured Information Standards (OASIS) (MacKenzie et al., 2006) states that a reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations or other concrete details.

Several DL frameworks (Gonçalves et al., 2004; Kahn and Wilensky, 2006) and reference models (Candela et al., 2007) have addressed specific problems in DLS architectural design and implementation. A discussion of some prominent reference models now follows.

2.3.1 Streams, Structures, Spaces and Societies

The Streams, Structures, Spaces and Societies (5S) framework is a unified formal theory for DLs. It is an attempt to define and easily understand the complex nature of DLs in a rigorous manner. The framework is based on formal definitions, and abstraction of five fundamental concepts—Streams,

Structures, Spaces, Scenarios and Societies (Gonçalves et al., 2004). The five concepts, together with their corresponding definitions and examples, are summarised in Table 2-1.

Table 2-1. Summary of key aspects of the 5S framework

Concept	Description	Examples
Streams	Streams represent a sequence of elements of an arbitrary type	Text, video, audio, software
Structures	Structures specify the organisation of different parts of a whole	Collection, document, metadata
Spaces	Spaces are sets of objects, with associated operations, that obey certain constants	User interface, index
Scenarios	Scenarios define details for the behaviour of services	Service, event, action
Societies	Societies represent sets of entities and the relationships among them	Community, actors, relationships, attributes, operations

In the context of the aims of DLs, Gonçalves et al. (Gonçalves et al., 2004) outline an association between 5S and some aims of a DLS, with Streams being aligned with the overall communication and consumption of information by end users; Structures supporting the organisation of information; Spaces dealing with the presentation and access to information in usable and effective ways; Scenarios providing the necessary support for defining and designing services; and Societies defining how a DL satisfies the overall information needs of end users.

However, Candela et al. (Candela et al., 2008) state that the 5S framework is very general-purpose and thus less immediate. The 5S framework is also arguably aimed at formalising the DL aspects, as opposed to prescribing specific design guidelines.

2.3.2 Kahn and Wilensky framework

This is a generic information system framework for distributed digital object services with digital objects as the main building blocks. The framework is based on an open architecture that supports large and distributed digital information services. Kahn and Wilensky (Kahn and Wilensky, 2006) describe the framework in terms of the fundamental aspects of an open and distributed infrastructure, and how the basic components in such an infrastructure support storage, accessibility and management of digital objects.

In addition to a high level conceptual description of such a distributed information system, the framework primarily focuses on the network-based aspects of such an infrastructure (Kahn and Wilensky, 2006). Specifically, an elaborate description of how digital objects should be accessed via a Repository Access Protocol (RAP) is outlined. The framework also proposes the use of a handle server infrastructure as a means for mapping registered digital objects.

In essence, the framework merely prescribes conventional methods for the unique identification, reliable location, and flexible access to digital objects.

2.3.3 DELOS reference model

The DELOS Network of Excellence on DLs[8] was a European Union co-funded project aimed at integrating and coordinating research activities in DLs. The DELOS working group published a manifesto that establishes principles that facilitate the capture of the full spectrum of concepts that play a role in DLs (Candela et al., 2007). The result of this project was a reference model—the DELOS DL reference model—comprising to a set of concepts and relationships that collectively attempt to capture various entities of the DL universe.

A fundamental part of the DELOS reference model is the DL Manifesto, that presents a DL as a three-tier framework consisting of a DL, representing an organisation; a DLS, for implementing DL services; and a Digital Library Management System (DLMS), comprising of tools for administering the DLS. Figure 2-5[9] shows the interaction among the three sub-systems.

The reference model further identifies six core concepts that provide a firm foundation for DLs. These six concepts—Content, User, Functionality, Quality, Policy and Architecture—are enshrined within the DL and the DLS. All concepts, with the exceptions of the Architecture concept, appear in the definition of the DL. The Architecture is, however, handled by the DLS definition (Candela et al., 2008).

The Architecture component, addressed by the DLS, is particularly important in the context of this research as it represents the mapping of the functionality and content on to the hardware and software components. Candela et al. (Candela et al., 2008) attribute the inherent complexity of DLs and the interoperability challenges across DLs as the two primary reasons for having Architecture as a core component.

Another important aspect of the reference model, directly related to this research, are the reference frameworks needed to clarify the DL universe at different levels of abstraction. The three reference development frameworks are: Reference Model, Reference Architecture, and Concrete Architecture. In the context of architectural design, the Reference Architecture is vital as it provides a starting point for the development of an architectural design pattern, thus paving the way for an abstract solution.

2.3.4 Summary

The motivation behind building both the reference models was largely influenced by the need to understand the complexity inherent in DLs. The idea of designing a DL architecture based on direct user needs is not taken into account in existing reference models, although the DELOS Reference Architecture does have a provision for the development of specific architectural design patterns. The DELOS Reference Architecture is in actual fact considered to be mandatory for the

[8] http://www.delos.info

[9] Permission to reproduce this image was granted by Donatella Castelli

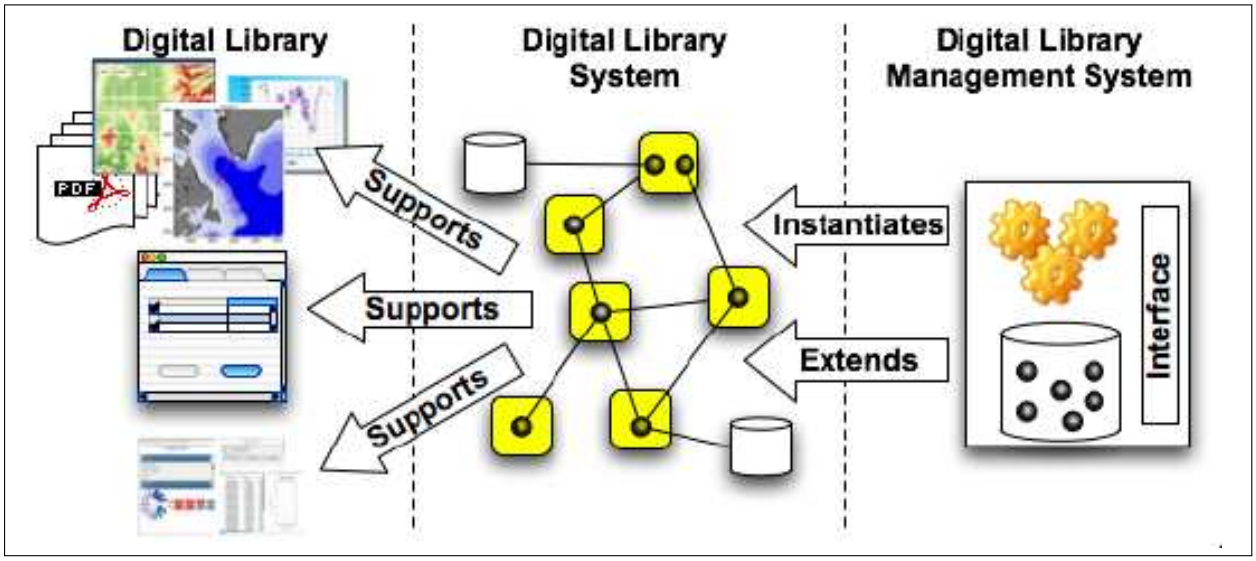


Figure 2-5. DL, DLS and DLMS: A three-tier framework

development of good quality DLSes, and for the integration and reuse of the system components.

2.4 Software platforms

There are a number of different DL software tools currently available. The ubiquitous availability of these tools could, in part, be as a result of specialised problems that these solutions are designed to solve. This section discusses seven prominent DL software platforms.

2.4.1 CDS Invenio

CDS Invenio, formally known as CDSware, is an open source repository software, developed at CERN[10] and originally designed to run the CERN document server[11]. CDS Invenio provides an application framework with necessary tools and services for building and managing a DL (Vesely et al., 2004).

The ingested digital objects' metadata records are internally converted into a MARC 21 — MARCXML— representation structure, while the actually fulltext bitstreams are automatically converted into PDF. This ingested content is subsequently accessed by downstream services via OAI service providers, email alerts and search engines (Pepe et al., 2005).

The implementation is based on a modular architecture. It is implemented using the Python Programming language, runs within an Apache/Python Web application server, and makes use of a MySQL backend database server for storage of metadata records.

[10] http://www.cern.ch
[11] http://cdsweb.cern.ch

2.4.2 DSpace

DSpace is an open-source repository software that was specifically designed for storage of digital research and institutional materials. The architectural design was largely influenced by the need for materials to be stored and accessed over long periods of time (Tansley et al., 2003).

The digital object metadata records are encoded using qualified Dublin Core—to facilitate effective resource description. Digital objects are accessed and managed via application layer services that support protocols such as OAI-PMH.

DSpace is organised into a three-tier architecture, composed of: an application layer; a business logic layer; and a storage layer. The storage layer stores digital content within an asset store—a designated area within the operating system's filesystem; or can alternatively use a storage resource broker. The digital objects —bitstreams and corresponding metadata records— are stored within a relational database management system (Smith et al., 2003; Tansley, Bass, and Smith, 2003). Furthermore software is implemented using the Java programming languages, and is thus deployed within a Servlet Engine. However, this architectural design approach arguably makes it difficult to recover digital objects in the event of a disaster since technical expertise would be required.

2.4.3 EPrints

EPrints is an archival software that designed to create highly configurable Web-based archives. The initial design of the software can be traced back to a time when there was a need to foster open access to research publications, and provides a flexible DL platform for building repositories (Gutteridge, 2002).

Eprints records are represented as data objects that contain metadata. The software's plugin architecture enables the flexible design and development of export plugins capable of converting repository objects into a variety of other formats. This technique effectively makes it possible for the data objects to be disseminated via different services—such as OAI data provider modules.

EPrints is implemented using Perl, runs within an Apache HTTP server and uses a MySQL database server backend to store metadata records. However, the actual files in the archive are stored on the filesystem.

2.4.4 ETD-db

The ETD-db digital repository software for depositing, accessing and managing ETD collections. The software is more oriented towards helping facilitate the access and management of ETDs.

The software was initially developed as is a series of Web pages and additional Perl scripts that interact with a MySQL database backend (*ETD-db: Home* 2012). However, the latest version—ETD 2.0—is a Web application, implemented using the Ruby on Rails Web application framework. This was done in an effort to handle ETD collections more reliably and securely. In addition, the

latest version is able to work with any relational database and can be hosted on any Web server that supports Ruby on Rails (Park et al., 2011).

2.4.5 Fedora Commons

Fedora is an open source digital content repository framework designed for managing and delivering complex digital objects (Lagoze et al., 2006).

The Fedora architecture is based on the Kahn and Wilensky framework (Kahn and Wilensky, 2006), discussed in Section 2.3.2, with a distributed model that makes it possible for complex digital objects to make reference to content stored on remote storage systems.

The Fedora framework is composed of loosely coupled services —implemented using the Java programming language— that interact with each other to provide the functionally of the Web service as a whole. The Web service functionalities are subsequently exposed via REST and SOAP interfaces.

2.4.6 Greenstone

Greenstone is an open source digital collection building and distributing software. The software's ability to redistribute digital collections on self-installing CD-ROMs has made it a popular tool of choice in regions with very limited bandwidth (Witten, Bainbridge, and Boddie, 2001).

The most recent version—Greenstone3 (Don, 2006)—is implemented in Java, making it platform independent. It was redesigned to improve the dynamic nature of the Greenstone toolkit and to further lower the potential overhead incurred by collection developers. In addition, it is distributed and can thus be spread across different servers. Furthermore, the new architecture is modular, utilising independent agent modules that communicate using single message calls (Bainbridge et al., 2004).

Greenstone uses XML to encode resource metadata records —XLinks are used to represent relationships between other documents. Using this strategy, resources and documents are retrievable through XML communication. Furthermore, indexing documents enables effective searching and browsing of resources.

The software operates within an Apache Tomcat Servlet Engine.

2.4.7 Omeka

Omeka is a Web-based publishing platform for publishing digital archives and collections (Kucsma, Reiss, and Sidman, 2010). It is standards-based and highly interoperable—it makes use of unqualified Dublin Core and is OAI-PMH compliant. In addition, it is relatively easy to use and has a very flexible design, which is customisable and highly extensible via the use of plugins.

Omeka is implemented using the PHP scripting language and uses MySQL database as a backend for storage of metadata records. However, the ingested resources—bitstreams— are stored on the filesystem.

2.4.8 Summary

Table 2-2 is a feature matrix of the digital libraries software discussed in this section.

Table 2-2. Feature matrix for some popular DL FLOSS software tools

		CDS Invenio	DSpace	EPrints	ETD-db	Fedora Commons	Greenstone	Omeka
Storage	**Complex object support**					X		
	Dublin Core support for metadata		X	X		X	X	X
	Metadata is stored in database	X	X	X	X	X	X	X
	Metadata can be stored on filesystem						X	
	Supports distributed repositories	X	X	X	X	X	X	X
Services	**Object relationship support**					X		X
	Extensible via plugins	X	X	X		X	X	X
	OAI-PMH complaint	X	X	X	X	X	X	X
	Platform independent		X	X		X	X	X
	Supports Web services		X			X	X	
	URI support(e.g. DOIs)		X			X		
Features	**Alternate accessibility (e.g. CD-ROM)**						X	
	Easy to setup, configure and use			X			X	X
	Handles different file formats	X	X	X		X	X	X
	Hierarchical collection structure		X	X		X	X	
	Horizontal market software	X	X	X		X	X	X
	Web interface	X	X	X	X	X	X	X
	Workflow support	X	X	X	X			

2.5 Minimalist philosophy

The application of minimalism in both software and hardware designs is widespread, and has been employed since the early stages of computing. The Unix operating system is perhaps one prominent example that provides a unique case of the use of minimalism as a core design philosophy, and

Raymond (Raymond, 2004) outlines the benefits, on the Unix platform, of designing for simplicity. This section discusses relevant architectures that were designed with simplicity in mind.

2.5.1 Dublin Core element set

The Dublin Core metadata element set defines a set of 15 resource description properties that are potentially applicable to a wide range of resources. One of the main goals of the Dublin Core element set is aimed at keeping the element set as small and simple as possible to facilitate the creation of resource metadata by non-experts (Hillmann, 2005).

Table 2-3. Simple unqualified Dublin Core element set

Element	Element Description
Contributor	An entity credited for making the resource available
Coverage	Location specific details associated to the resource
Creator	An entity responsible for creating the resource
Date	A time sequence associated with the resource life-cycle
Description	Additional descriptive information associated to the resource
Format	Format specific attributes associated with the resource
Identifier	A name used to reference the resource
Language	The language used to publish the resource
Publisher	An entity responsible for making the resource available
Relation	Other resource(s) associated with the resource
Rights	The access rights associated with the resource
Source	The corresponding resource where the resource is derived from
Subject	The topic associated to the resource
Title	The name of the resource
Types	The resource type

The simplicity of the element set arises from the fact that the 15 elements form the smallest possible set of elements required to describe a generic resource. In addition, as shown in Table 2-3, the elements are self explanatory, effectively making it possible for a large section of most communities to make full use of the framework. Furthermore, all the elements are repeatable and at the same time optional. This flexibility of the scheme is, in part, the research why it is increasingly becoming popular.

2.5.2 Wiki software

Wiki software allows users to openly collaborate with each other through the process of creation and modification of Web page content (Leuf and Cunningham, 2001). The success of Wiki software is, in part, attributed to the growing need for collaborative Web publishing tools. However, the simplicity in the way content is managed, to leverage speed, flexibility and easy of use, is

arguably the major contributing factor to their continued success. The strong emphasis on simplicity in the design of Wikis is evident in Cunningham's original description: "The simplest online database that could possibly work" (*What is Wiki* 1995; Leuf and Cunningham, 2001).

2.5.3 Extensible markup language

XML is a self-describing markup language that was specifically designed to transport and store data. XML provides a hardware- and software-independent mode for carrying information, and was design for ease of use, implementation and interoperability from the onset. This is in fact evident from the original design goals that, in part, emphasised for the language to be easy to create documentations, easy to write programs for processing the documents and straightforwardly usable over the Internet (Bray et al., 2008).

XML has become one of the most commonly used tool for transmission of data in various applications due to the following reasons.

- Extensibility through the use of custom extensible tags
- Interoperability by being usable on a wide variety of hardware and software platforms
- Openness through the open and freely available standard
- Simplicity of resulting documents, effectively making them readable by machines and humans

The simplicity of XML particularly makes it an easy and flexible tool to work with, in part, due to the fact that the XML document syntax is composed of a fairly minimal set of rules. Furthermore, the basic minimal set of rules can be expanded to grow more complex structures as the need arises.

2.5.4 OAI protocol for metadata harvesting

The OAI-PMH is a metadata harvesting interoperability framework (Lagoze et al., 2002b). The protocol only defines a set of six request verbs, shown in Table 2-4, that data providers need to implement. Downstream service providers then harvest metadata as a basis for providing value-added services.

Table 2-4. OAI-PMH request verbs

Request Verb	Description
GetRecord	This verb facilitates retrieval of individual metadata records
Identify	This verb is used for the retrieval of general repository information
ListIdentifiers	This verb is used to harvest partial records in the form of record headers

(Continued on next page)

Table 2-4. (continued)

Request Verb	Description
ListMetadataFormats	This verb is used to retrieve metadata formats that are supported
ListRecords	This verb is used to harvest complete records
ListSets	This verb is used to retrieve the logical structure defined in the repository

The OAI-PMH framework was initially conceived to provide a low-barrier to interoperability with the aim of providing a solution that was easy to implement and deploy (Lagoze and Sompel, 2001). The use of widely used and existing standards, in particular XML and Dublin Core for encoding metadata records and HTTP as the underlying transfer protocol, renders the protocol flexible to work with. It is increasingly being widely used as an interoperability protocol.

2.5.5 Project Gutenberg

Project Gutenberg[12] is a pioneering initiative, aimed at encouraging the creation and distribution of eBooks, that was initiated in 1971 (*About Gutenberg* 2011). The project was the first single collection of free electronic books (eBooks) and its continued success is attributed to its philosophy (Hart, 1992), where minimalism is the overarching principle. This principle was adopted to ensure that the electronic texts were available in the simplest, easiest to use forms; independent of the software and hardware platforms used to access the texts.

2.5.6 Summary

This section has outlined, through a discussion of some prominent design approaches, how simplicity in architectural designs can be leveraged and result in more flexible systems that are subsequently easy to work with. In conclusion, the key to designing easy to use tools, in part, lies in identifying the least possible components that can result in a functional unit and subsequently add complexity, in the form of optional components, as need arises. Minimalist designs should not only aim to result in architectures that are easier to extend, but also easier to work with.

2.6 Data storage schemes

The repository sub-layer forms the core architectural component of a typical digital library system and more specifically, it is composed of two components: a bitstream store and a metadata store, responsible for storing digital content and metadata records respectively. As shown in Table 2-2, DLSes are generally implemented in such a manner that digital content is stored on the file system, whilst the metadata records are almost always housed in a relational database.

[12] http://www.gutenberg.org

This section discusses three prominent data storage solutions that can potentially be integrated within the repository sub-layer for metadata storage. The focus is to assess their suitability for integration with DLSes.

2.6.1 Relational databases

Relational databases have stood the test of time, having been around for decades. They have, until recently, been the preferred choice for data storage. There are a number of reasons (see Elmasri and Navathe, 2008, chap. 3) why relational databases have proved to be a popular storage solution, and these include:

- The availability of a simple, but effective query language—SQL— capable of retrieving multifaceted views of data
- Support for Data model relationships via table relations
- Transaction support through ACID[13] properties
- Support for data normalisation, thus preventing redundancy

Relational databases are, however, mostly suitable for problem domains that require frequent retrieval and update of relatively small quantities of data.

2.6.2 NoSQL databases

The large-scale production of data (Gantz et al., 2008), coupled with the now prevalent Big Data[14], has resulted in a profound need for data storage architectures that are efficient, horizontally scalable, and easier to interface with. As a result, NoSQL databases recently emerged as potential alternatives to relational databases. NoSQL databases are non-relational databases that embrace schemaless data, are capable of running on clusters, and generally trade off consistency for other properties such as performance (see Sadalage and Fowler, 2012, chap. 1).

NoSQL database implementations are often categorised based on the manner in which they store data, and typically fall under the categories described in Table 2-5.

Table 2-5. Data model categories for NoSQL database stores

Data Model	Description
Column-Family Stores	Data is stored with keys mapped to values grouped into column families
Document Stores	Data is stored in self-describing encoded data structures
Graph Stores	Data is stored as entities with corresponding relationships between entities

(Continued on next page)

[13] Atomicity, Consistency, Isolation, Durability

[14] http://www-01.ibm.com/software/data/bigdata

Table 2-5. (continued)

Data Model	Description
Key-Value Stores	Hash table with unique keys and corresponding pointer to blobs

NoSQL databases are highly optimised for retrieve and append operations and, as a result, there has recently been an increase in the number of applications that are making use of NoSQL data stores. However, the downside of NoSQL databases is that they cannot simultaneously guarantee data consistency, availability and partition tolerance; as defined in the CAP theorem (Gilbert and Lynch, 2002).

2.6.3 Filesystems

File systems are implemented by default in all operating systems, and provide a persistent store for data. In addition, they provide a means to organise data in a manner that facilitates subsequent retrieval and update of data.

Native file systems have, in the past, not generally been used as storage layers for enterprise applications, in part, due to the fact that they do not provide explicit support for transaction management and fast indexing of data. However, the emergence of clustered environments has resulted in robust and reliable distributed file system technologies such as Apache Hadoop (Borthakur, 2007) and Google File System (Ghemawat, Gobioff, and Leung, 2003).

The opportunities presented by traditional file systems, and in particular their simplicity, efficiency and general ease of customisation make them prime candidates for storage of both digital content and metadata records. In addition, the use of flat files, and more specifically text files, for storage of metadata records could further complement and simplify the digital library repository sub-layer. Incidentally, Raymond (see Raymond, 2004, chap. 5) highlights a number of advantages associated with using text files, and further emphasises that designing textual protocols ultimately results in future-proof systems.

In general, there are a number of real-word application whose data storage implementations take advantage of file systems. Some notable example implementations of both digital libraries specific tools and general purpose tools are outlined below.

BagIt file packaging format The BagIt File Packaging Format specification (Boyko et al., 2012) defines a hierarchical file packaging format suitable for exchanging digital content. The BagIt format is streamlined for disk-based and network-based storage and transfer. The organisation of bags is centred on making use of file system directories as bags, which at a minimum contain: a data directory, at least one manifest file that lists data directory contents, and a bagit.txt file that identifies the directory as a bag.

DokuWiki DokuWiki is a PHP based Wiki engine, mainly aimed at creating documentation, that is standards compliant and easy to use (Gohr, 2004). The storage architecture of DokuWiki principally makes use of the filesystem as its data store, with application data files stored in plain text files. This design strategy ensures that data is accessible even when the server goes down, and at the same time facilities backup and restore operations through the use of basic server scripts and FTP/sFTP.

Git Git is a distributed version control system that functions as a general tool for filesystem directory content tracking, and is designed with a strong focus on speed, efficiency and real-world usability on large projects (see Chacon, 2009, chap. 1), to attain three core functional requirements below.

- Store generic content
- Track content changes in the repository
- Facilitate a distributed architecture for the content

Git is internally represented as a duplex data structure that is composed of a mutable index for caching information about the working directory; and an immutable repository. The Git object storage area is a Directed Acyclic Graph that is composed of four types of objects—blob objects; tree objects; commit objects; and tag objects (*Git for Computer Scientists* 2010). In addition, the repository is implemented as a generic content-addressable filesystem with objects stored in a simple key-value data store (see Chacon, 2009, chap.9).

Pairtrees for collection storage Pairtree is a file system convention for organising digital object stores, and has the advantage of making it possible for object specific operations to be performed by making use of native operating system tools (Kunze et al., 2008).

2.6.4 Summary

There are numerous available data storage options, and it is important to understand the varying options to fully identify the ones most applicable to specific problem domains. It is generally not always the case that a definitive storage solution is arrived at, however, a data model that better matches the kind of data storage and retrieval requirements should be the primary deciding factor. Table 2-6 is a summarised comparative matrix outlining the three storage solutions discussed in this section.

It is that it is generally not always necessary to use an intermediate data management infrastructure, and in some cases, it may in all actuality be desirable not to use one at all; as is the case with the real world applications described in Section 2.6.3.

Table 2-6. Comparative matrix for data storage solutions

	File Systems	**RDBMS**	**NoSQL**
Use Cases	Miscellaneous	Relational data	Large-scale data
Data Format	Heterogeneous data	Structured data	Unstructured data
Transaction Support	Simple Locking	ACID compliant	CAP theorem support
Indexing	Optional	Available	Available
Scalability	Horizontal; Vertical	Vertical	Horizontal
Replication	Partial support	Explicit support	Explicit support

2.7 Design decisions

Software architectures provide an overview of a software system's components, and the relationships and characteristics that exist between the various components (Lee and Kruchten, 2007). The architectures are initially conceived as a composition of the general design, influenced by a corresponding set of design decisions (Kruchten et al., 2005). The design decisions form a fundamental part of the architectural design process, by guiding the development of the software product, as they help ensure that the resulting product conforms to desired functional and non-functional requirements.

There are two prominent methods—design rationale and formalised ontological representation—used to capture design decisions (Lee and Kruchten, 2007). The design rationale provides a historical record, in form of documentation, of the rationale used to arrived at a particular design approach (Lee and Lai, 1991), and typically makes use of techniques such as Issue-Based Information Systems (IBIS) (Conklin and Yakemovic, 1991) and "Questions, Options and Criteria" (QOC) (MacLean et al., 1991). The formalised ontological representation method on the other hand makes use of an ontological model for describing and categorising the architectural design decisions (Kruchten, 2004).

There are a number of benefits of explicitly capturing and documenting design decisions, the most significant one being that they help in—ensuring the development of the desired product. In the case of domain-specific products, they form a crucial role of ensuring that the resulting solution directly conforms to the solution space it is meant to operate within.

2.8 Summary

This chapter discussed background information that forms the basis for this research. Section 2.1 discussed DLs, through elaborate high level definitions, complemented with examples of varying

application domains within which contemporary DLSes are utilised. Core fundamental concepts associated to DL were also discussed in 2.2.

Some prominent DL frameworks were presented in Section 2.3, followed by FLOSS DL software tools in Section 2.4; revealing that the varying frameworks and architectural designs are largely as a result of the different problems for which solutions were sought. However, there are core features that are common to most of the proposed solutions. It could thus be argued that existing solutions may not be be suitable for certain environments, and as such simpler alternative architectural designs may be desirable. A culmination of the argument for utilising simpler architectural designs manifested in the discussion of prominent designs that used simplicity as the core criterion in Section 2.5.

In addition, the repository sub-layer was highlighted as the component that forms the core of digital libraries in Section 2.6, and a further discussing of potential storage solutions that can be used for the storage of metadata records then followed. Traditional file systems have been identified as contenders of the more generally accepted relational databases and now common place NoSQL databases, for the storage of metadata records.

Furthermore, a discussion of two major general approaches followed when arriving at software design decisions were presented in 2.7.

Chapter 3

Design principles

This chapter details the systematic process that was followed in order to derive the guiding principles that can potentially simplify the design of DL services, effectively making then easier to work with.

The chapter is organised as follows: Section 3.1 outlines the rationale behind conducting this exploratory study; Section 3.1.2 introduces and describes the research method that was employed during this phase of the research; Section 3.3 details the process that was followed to collect and analyse the data; and finally, Section 3.4 concludes the chapter.

3.1 Research perspective

3.1.1 Prior research observations

In our earlier work linked to this research, some issues that hinder ubiquitous access to information and widespread preservation in Africa have been highlighted (Suleman, 2008). A number of potential solutions to the issues raised have in the recent past also been presented, and take the form of lightweight systems (Suleman, 2007; Suleman et al., 2010) with simplicity as the key criterion.

However, the proposed solutions were solely based on specific user requirements. The significance of prior work stems from the fact that they provided this research with working hypotheses, which take the form of a set of observable facts, that helped set the stage for the exploratory study.

3.1.2 Research questions

The primary research question for this research, described in Section 1.3, seeks to investigate the feasibility of implementing DL services that are based on simple architectures. In order to better understand the simplicity of services, a secondary research question, which was the main driving factor for the exploratory study, was formulated as outlined below.

- How should simplicity for DL storage and service architectures be defined?

The overall aim of the exploratory study was two-fold: firstly, it served to guide the overall direction of the research, and secondly, it was aimed at understanding contemporary DL design in such a way as to be able to better prescribe an alternative design approach that might result in simpler DL tools and services.

In order to obtain a reliable and comprehensive understanding of the desired result, a qualitative study was conducted using a Grounded Theory approach.

3.1.3 Summary

This section has highlighted prior work related to this research that helped set the stage for the exploratory study. The section also outlined how the exploratory study fits into the overall aims of the research by outlining the rationale and significance of the study. In the subsequent section, the research methods used during the exploratory study are discussed.

3.2 Research methods

3.2.1 Grounded theory

Grounded Theory is a research method that provides a technique for developing theory iteratively from qualitative data. The goal of Grounded Theory is to generate a theory that accounts for a pattern of behaviour that is relevant for those involved (Glaser, 1978). Grounded Theory attempts to find the main concern of the participants and how they go about resolving it, through constant comparison of data at increasing levels of abstraction and has also been described as "a general pattern for understanding" (Glaser, 1992).

The Grounded Theory method generally revolves around a series of five steps, as outlined below.

Grounded theory process

Step 1 Data collection

This step uses a method appropriate to the research context to elicit information from selected participants. Typical methods include conducting semi-structured interviews.

Step 2 Data analysis

The data analysis step forms the core of grounded theory and generally involves the use of a constant comparative method to generate and analyse data.

Step 3 Memoing

Memoing, as the name suggests, involves writing theoretical memos to identify relationships between different patterns of the data.

Step 4 Sorting

The sorting step takes the form of arranging all memos once the data collection becomes saturated. The outcome of this results in a theory describing how the identified categories relate to the core category.

Step 5 Theoretical coding

The data collected is divided into segments to identify categories or themes. The categorised data is then further examined to identify properties common to each of the categories.

Grounded theory was selected as the primary research method for the exploratory study due to the following reasons:

- It is primarily aimed at theory generation, focusing specifically on generating theoretical ideas, explanations and understanding of the data.
- It is useful when trying to gain a fresh perspective of a well-known area.
- It has proven to be a successful method for exploring human and social aspects.
- It is by far one of the most common/popular analytic technique in qualitative analysis.
- It is arguably intuitive.

3.2.2 Analytic hierarchy process

The Analytic Hierarchy Process (AHP) is a theory of measurement through pairwise comparisons that relies on judgement of experts to derive priority scales (Saaty, 2008). A pairwise comparison is a problem-solving technique that allows one to determine the the most significant item among a group of items. The overall process is driven by scales of absolute judgement that represent how much more an element dominates another with respect to a given attribute. The pairwise comparison method involves following a series of steps and is outlined in Section 3.2.2.

Pairwise comparisons method

The method of pairwise comparisons ensures that for a given set of elements or alternatives, each candidate element is matched head to head with the other candidates and is performed by decomposing decisions (Saaty, 2008) into the steps outlined below.

Step 1 Define the criteria to be ranked.

The criteria identified are influenced by the overall objectives and form the basis of the comparative analysis.

Step 2 Arrange the criteria in an $N \times N$ matrix.

In essence, each element in a given set a of N elements is compared against other alternatives in the set as shown in Table 3-1. The total number of pairwise comparisons can thus be computed using equation:

$$\frac{N(N-1)}{2}$$

Table 3-1. An $N \times N$ pairwise comparisons matrix

	H	**G**	**F**	**E**	**D**	**C**	**B**	**A**
A	X	X	X	X	X	X	X	
B	X	X	X	X	X	X		
C	X	X	X	X	X			
D	X	X	X	X				
E	X	X	X					
F	X	X						
G	X							
H								

Step 3 Compare pairs of items.

Each criterion is compared again other alternatives to determine the relative important of the characteristic.

Step 4 Create the ranking of items.

A ranking system is created based on the relative occurrence of each element in the matrix.

The use of pairwise comparisons was particularly useful in the research context as the method is ideal for ranking a set of decision-making criteria and rate the criteria on a relative scale of importance.

3.2.3 Summary

This section has described the two primary research methods that were used during the exploratory phase of this research. The combined effect of using the two methods is appropriate as the ex-

ploratory study involved a series of qualitative phases. The details of the study are outlined in Section 3.3.

3.3 General approach

3.3.1 Data collection

A meta-analysis involving a total of 12 software applications was systematically conducted to facilitate the compilation of a comprehensive and inclusive set of principles. The set of tools comprised six DL software applications and six non-DL software applications. The selection of the six DL software was done on the basis of popularity as depicted on OpenDOAR[1]. Table 3-2 outlines the 12 candidate tools that were considered.

The relevant software attributes that may have influenced the design decisions of the applications were then identified. The pairwise comparisons method, outlined in Section 3.2.2 was then used as the constant comparison method during the data analysis stage. The data analysis stage is discussed in more detail in Section 3.3.2

Table 3-2. Software applications used for pairwise comparisons

Application	Category	Description
DSpace[2]	DL software	A general digital asset management software
EPrints[3]	DL software	A general digital repository software package
ETD-db[4]	DL software	An electronic thesis and dissertation software package
FedoraCommons[5]	DL software	A general digital object repository framework
Greenstone[6]	DL software	A general digital collection management software
CDS Invenio[7]	DL Software	A general document repository software package
Facebook[8]	Non DL software	A free social network portal/Website
Gmail[9]	Non DL software	A free email messaging hosted-service platform
MixIt[10]	Non DL software	A free instant messaging Web application
Moodle[11]	Non DL software	A free e-learning management platform

(Continued on next page)

[1] http://www.opendoar.org
[2] http://www.dspace.org
[3] http://www.eprints.org
[4] http://scholar.lib.vt.edu/ETD-db
[5] http://fedora-commons.org
[6] http://www.greenstone.org
[7] http://invenio-software.org
[8] http://www.facebook.com
[9] https://mail.google.com
[10] http://www.mixit.com
[11] http://moodle.org

Table 3-2. (continued)

Application	Category	Description
Ushahidi[12]	Non DL software	An information collection and visualisation platform
WordPress[13]	Non DL software	A standalone blogging software package

3.3.2 Data analysis

The set of all possible software attributes that can potentially influence design decisions of software applications were identified and arranged based on whether they were specific to the two sets of software applications—Digital Library software and non-DL software—or both sets. Table 3-3 shows the software attributes, that were considered, as pertains to whether they affect DL software, non-DL software, or both.

Table 3-3. Software attributes considered in pairwise comparisons

	DL	Non-DL	Both
Digital content	X		
Media types	X		
Metadata objects	X		
Data access			X
Information structure	X		
Core language			X
Content delivery			X
Deployment platform			X
Software dependencies			X
Flexibility			X
Preservation strategy	X		
Extensibility			X
Standardisation			X
Interoperability			X
Ease of installation			X
Objects accessibility	X		
Objects naming scheme	X		
Hosting			X
Scalability			X
Reliability			X
Usability			X
Mobile friendly			X

[12] http://www.ushahidi.com
[13] http://wordpress.org

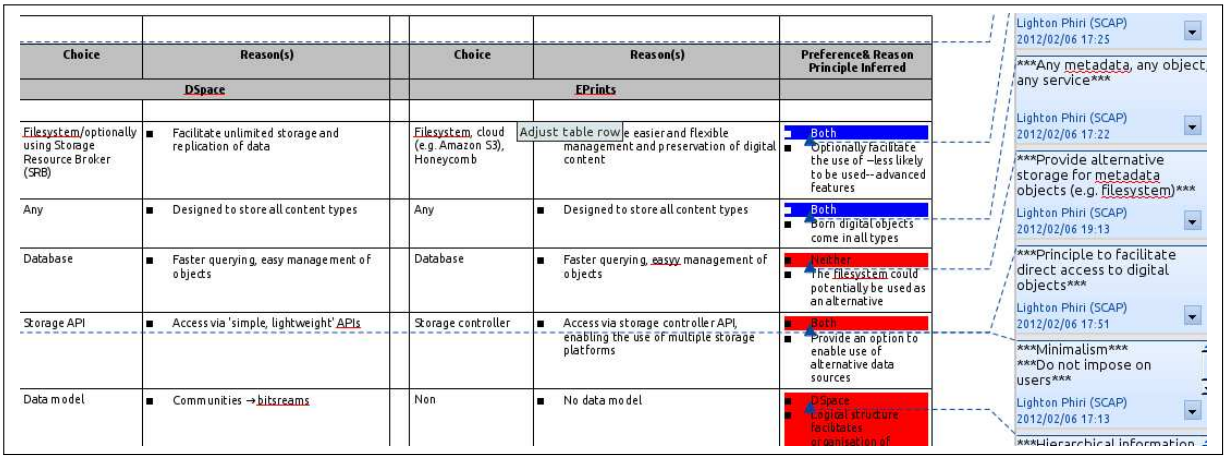


Figure 3-1. Screenshot showing an excerpt of the grounded theory memoing process

Open coding (Glaser, 1992) was used during the data analysis process, and a head-to-head pairwise comparison was then performed on each of the 12 applications against the other alternatives using the pairwise comparisons method procedure described in Section 3.2.2. All in all, a total of 66 pairwise comparisons, derived using the equation in Section 3.2.2, were conducted.

The Memoing process, for each of the 66 comparisons, involved identifying design choice for each software attribute and the possible corresponding design rationale. All possible potential design decisions that could be applicable to the design of simple and minimalistic architectures were subsequently identified. Figure 3-1 shows an excerpt of the memoing process.

3.3.3 Design principles

The major outcome of the exploratory study is a set of eight guiding design principles for simple and minimalistic architectures of digital libraries tools and/or services. It is premised that DL software designed and implemented based on these guiding principles could ultimately be easy to use and maintain in the long run. The design principles are as follows:

Principle 1. Hardware and/or software platform independence

Description It should be possible to operate tools and services on a wide variety of hardware and software platforms. The rationale behind this principle is to ensure that the least possible cost associated to technological infrastructure is incurred during the collection management life-cycle.

Discussion The preservation life-cycle of digital objects is an on-going process that typically involves the management of digital content and its associated representational information. The cost implications of long-term digital preservation is a crucial task for both small and large-scale preservation projects (Beagrie et al., 2002). However, the vast majority of organisations involved in the curation and preservation of digital information usually do not have adequate funding to

support this process. In addition, a number of such organisations, in particular heritage organisations, do not have sustainable funding models to ensure the on-going process of managing digital objects.

A reduction in the cost associated to the collection management process could be archived in various ways including, but not limited to the following:

- Designing tools that require minimal technical expertise to manage
- Designing tools capable of being run on popular of operating systems
- Designing tools capable of being operated on hardware platforms with minimal specifications

Principle 2. Heterogeneous object, metadata and service integration

Description There should be explicit support for integration of any digital object type, metadata format or new service.

Discussion The proliferation of both born-digital and digitised information has given rise to various data formats and a corresponding increase in the number of metadata standards, as discussed in Section 2.2.3. In addition, there is a growing demand for DL services in order to facilitate ubiquitous access to information.

Due to the aforementioned, it is imperative that the designed of digital library tools be flexible enough to accommodate heterogeneous objects, metadata and services. In a nutshell, the design should be based on a "one size fits all" approach.

Principle 3. Support for community and international standards

Description The design of tools and services should take into account community-based standards and international standards in order to facilitate interoperability.

Discussion The increase in the amount of digital content generated and made available publicly has brought about a need to standardise processes in the digital curation workflow. Section 2.2.4 outlines the important role that standards play and also discusses some of the popular DL standards.

Incorporating standards in the initial stages of the design process would effectively ensure that the resulting DL services becomes interoperable with other external services. It also makes it easier for service to be customised.

Principle 4. Flexible design to facilitate extensibility

Description The design should be flexible enough to enable end users to adapt the tools and services to their own needs.

Discussion Digital curation is slowly becoming a ubiquitous process, and DLs are increasingly being used in a wide array of application domains—example application domains are highlighted in Section 2.1.2.

The services offered by these different application domains varying and it is imperative that the overall design be flexible enough to facilitate customisation and extensibility.

Principle 5. Minimalist design approach

Description There should be minimal use of external software components in order to simplify the overall design. This would arguably result in tools that are easier to manage.

Discussion The design of services should, at a minimum, only be composed of the least number of components that are required for it to function. Auxiliary external components should be made optional, making them available only when required.

In addition, mandatory components should be critically analysed to ensure that they make use of simplest possible solutions and/or technologies.

Principle 6. Simplified preservation process

Description The preservation process should be simplified as much as possible to make it possible to easily migrate digital content.

Discussion The preservation lifecylce is an on-going process that requires dedicated staff. The majority of contemporary DL services require technology experts to perform the routine preservation tasks.

The overall design should thus be made as simple as possible so that novice users are able to perform the most basic of preservation tasks.

Principle 7. Structured Organisation of Data

Description There should be explicit support for hierarchical logical organisation of information.

Discussion The majority of data that is curated and made accessible publicly necessitates the logical organisation of information to facilitate relationships that might exist between different data views. In addition, data consumers usually visualise information using varying logical views.

The design should thus explicitly support the logical organisation of information, and optionally make it flexible enough for users to define the desired logical views and structures.

Principle 8. Design for least possible resources

Description There should be support for access to digital collections in environments with resource constraints.

Discussion One of the motivating factors, outlined in Section 1.1, behind this research was the unavailability of DL tools that can effectively operate in resource constrained environments. This is still a growing need for most environments in developing countries, such as those found in Africa.

The design of DL services should thus be based on the least possible resources to enable resulting service operate in environments with limited resources.

3.3.4 Summary

This section discussed the procedure that was followed to derived a set of design guiding principles that, when employed during the design of digital library services, may potentially result in simpler services. Grounded Theory was used as the overarching method during the derivation process and Table 3-4 shows a summary of how the Grounded Theory steps were undertaken.

Table 3-4. Grounded theory general approach

Stage	Description
Data Collection	A meta-analysis review of 12 software applications was conducted
Data Analysis	Pairwise comparisons were used at the constant comparative method
Memoing	Memos were created using a general note taking process
Sorting	Arranged conceptual levels based on meta-level of attribute being investigated
Coding	The coding process took place in tandem with the data collection process and open coding was used

3.4 Summary

This chapter discussed the derivation process of a set of design principles applicable for the design of simple DL services which can easily be operated in resource constrained environments. The development of applications for resource constrained environments requires careful consideration and eliciting these requirements during the early stages of the design process may ensure that the resulting services become tailored for such domains.

In summary, all the design principles were derived with simplicity and minimalism and the key criterion.

Chapter 4

Designing for simplicity

In Chapter 3, a set of design principles, and the systematic approach used to derived them was outlined. The derived design principles can be applied during the design of the different components of a DLS—user interface, repository and service layer.

In this chapter, a prototype generic simple repository design, based on the derived design principles, is outlined.

4.1 Repository design

4.1.1 Design decisions

In Section 2.7, the significance of software design decisions were outlined; in addition prominent methods used to capture design decisions were highlighted. The design decisions associated with the architectural design of the repository sub-layer were arrived by taking into account the principles derived during the exploratory study (see Chapter 3). Tables 4-1, 4-2, 4-3 and 4-4 outline the detailed design decisions applied to design the repository.

Table 4-1. Simple repository persistent object store design decision

Element	**Description**
Issues	Principles 1, 2, 6 and 8
Decision	Store bitstreams on the local operating system filesystem
Assumptions	None
Alternatives	Store bitstreams as blobs in a database; store bitstreams in the cloud
Rationale	Backup and migration tasks associate to repository objects can be potentially simplified; operating system commands can be used to perform repository management tasks
Implications	None –most conventional tools and services use the same approach
Notes	None

Table 4-2. Simple repository metadata storage design decision

Element	Description
Issues	Principles 1, 2, 5, 6 and 8
Decision	Native operating system filesystem used for metadata storage
Assumptions	None
Alternatives	Relational database; NoSQL database; embed metadata into digital objects
Rationale	Storing metadata records in plain text files ensures platform independence; complexities introduced by alternative third-party storage solution avoided through the use of native filesystem
Implications	No standard method for data access (e.g. SQL); Transaction process support only available via simple locking; non-availability of complex security mechanisms
Notes	None

Table 4-3. Simple repository object naming scheme design decision

Element	Description
Issues	Principle 5
Decision	Use actual object name as unique identifier
Assumptions	Native operating systems
Alternatives	File hash values; automatically generated identifiers
Rationale	Native operating systems ensure file naming uniqueness at directory level. In addition, it is a relatively simpler way of uniquely identifying objects as object naming control is given to end users, rather than imposing it on them
Implications	Object integrity has a potential to be compromised; objects could potentially be duplicated by simply renaming them
Notes	None

Table 4-4. Simple repository object storage structure design decision

Element	Description
Issues	Principles 6 and 7
Decision	Store bitstreams alongside metadata records –at the same directory level on the filesystem; filesystem directory to be used as container structures for repository objects
Assumptions	The other sub-layers of the DLS have read, write and execute access to the repository root node
Alternatives	Separate storage locations for bitstreams and metadata records

(Continued on next page)

Table 4-4. (continued)

Element	Description
Rationale	Storing bitstreams and corresponding metadata records alongside each other could ultimately make potential migration processes easier; container structures could potentially make it easier to move repository objects across different platforms
Implications	None
Notes	None

4.1.2 Architecture

The architectural design is centred around designing a simple repository which at a bare minimum is capable of facilitating the core features of a DLS—long term preservation and ease of access to digital objects.

Table 4-5. Simple repository component composition

Component	File Type	Description
Container Object	Directory	Structure used to store digital objects
Content Object	Regular file	Content/bitstreams to be stored in the repository
Metadata Object	Regular file	XML-encoded plain text file for storing metadata records

The repository design is file-based and makes use of a typical native operating system filesystem as the core infrastructure. Table 4-5 shows the main components that make up the repository sub-layer, with all the components residing on the filesystem, arranged and organised as normal operating system files—regular files and/or directories—as shown in Figure 4-1.

As shown in Figure 4-1, a typical DLS repository would be located in an application accessible base root directory node, and is composed of two types of digital objects—Container Objects and Content Objects—both of which are created and stored within the repository with companion Metadata Objects that store representational information associated with the object. Figure 4-2 illustrates how Container and Content objects are stored on a typical filesystem.

Container Objects can be recursively created within the root node as the repository scales, and exhibit an interesting characteristic of a enabling the creation of additional Container Objects within them. As shown in Figure 4-3, the Metadata Object associated with Container Objects holds information that uniquely identifies the object; optionally describe the object in more detail, including relationships that might exist with other objects within the repository; and a detailed log of objects contained within it—the manifest.

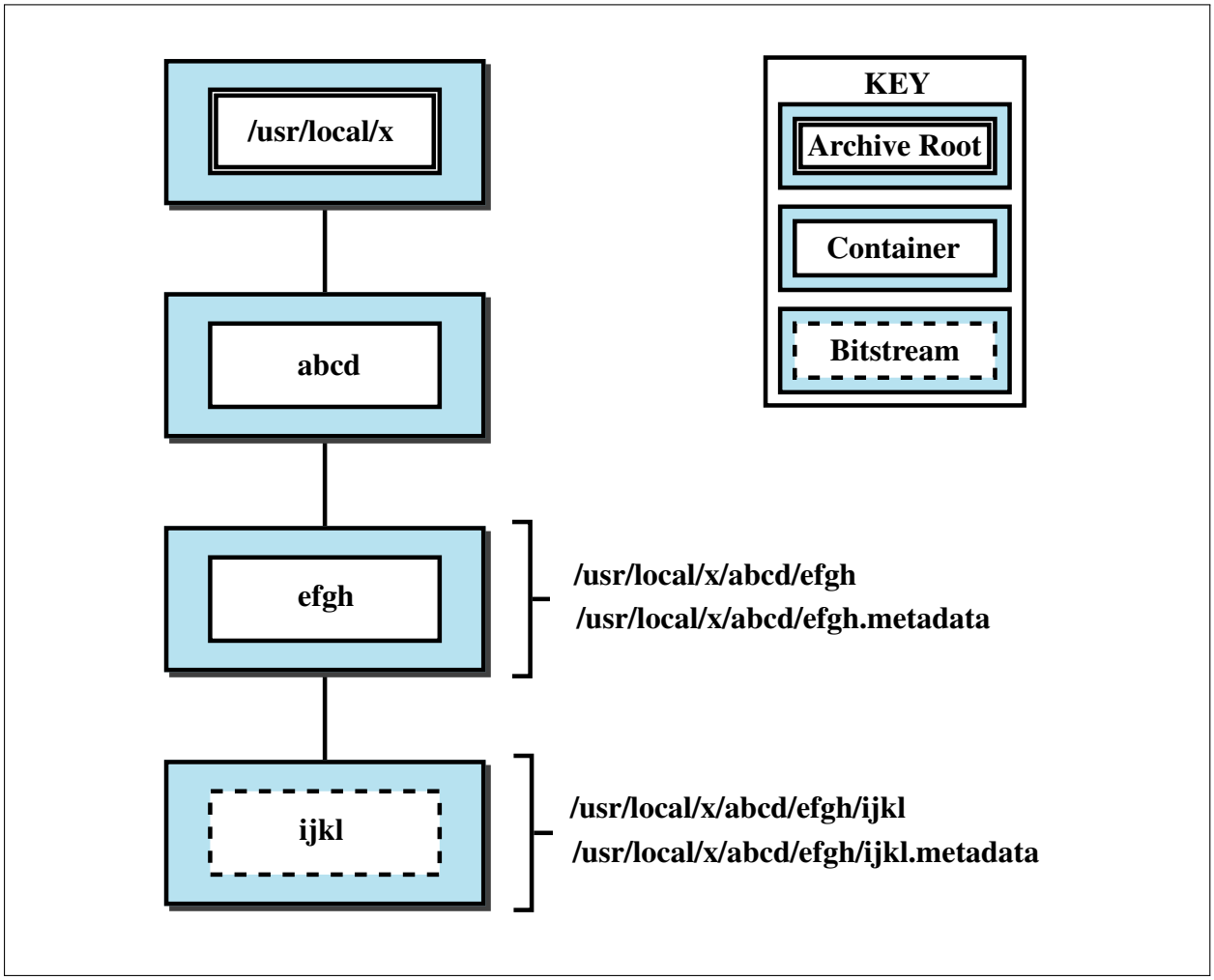


Figure 4-1. Simple repository object structure

Content Objects represent digital objects—typically bitstreams—to be stored within the repository. As shown in Figure 4-4, the representational information stored in the Metadata Objects associated with Content Objects is similar to that of Container Objects, with the exception of manifest related information.

4.1.3 Summary

In this chapter, the design of a prototype simple repository sub-layer was outlined through the mapping of design decisions and principles derived in Chapter 3.

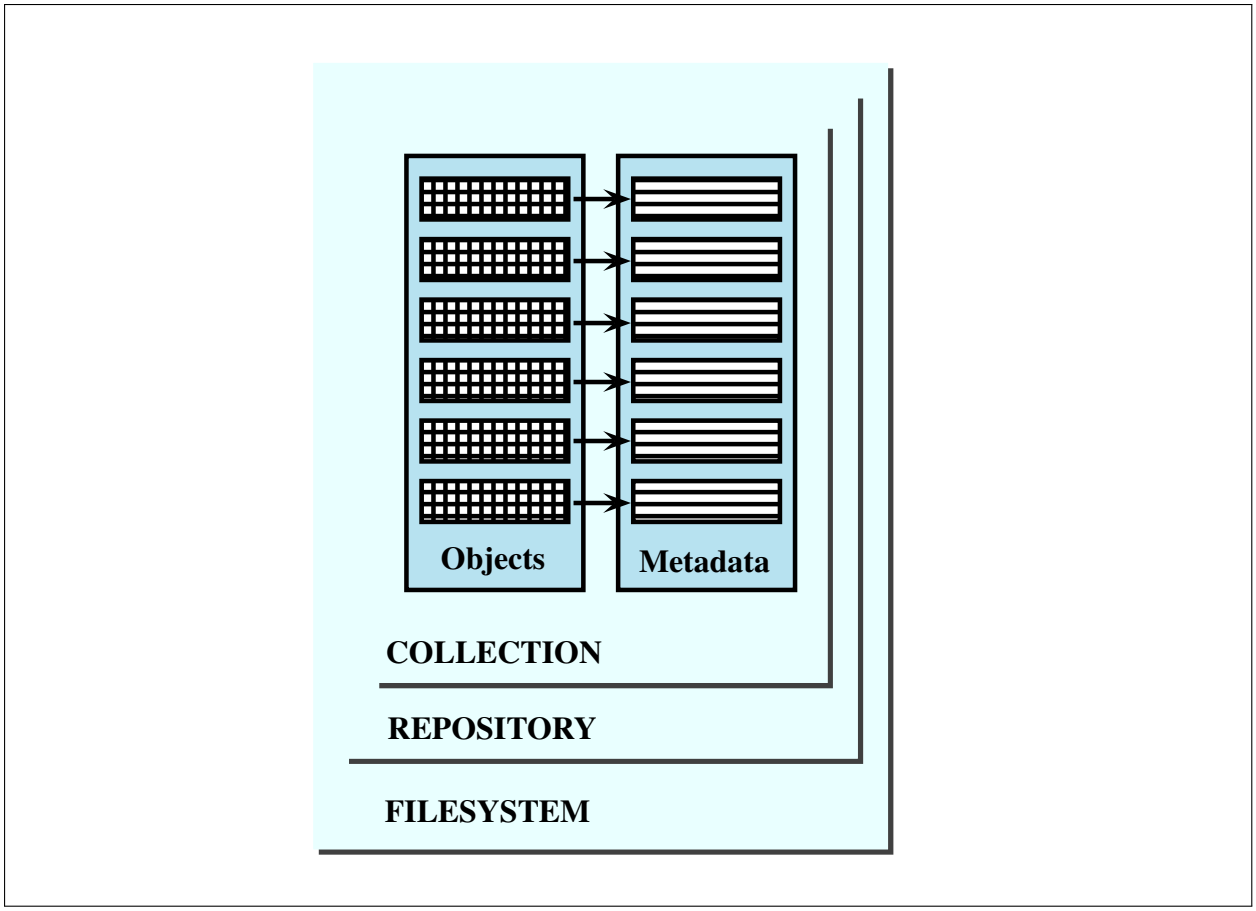


Figure 4-2. Simple repository object structure

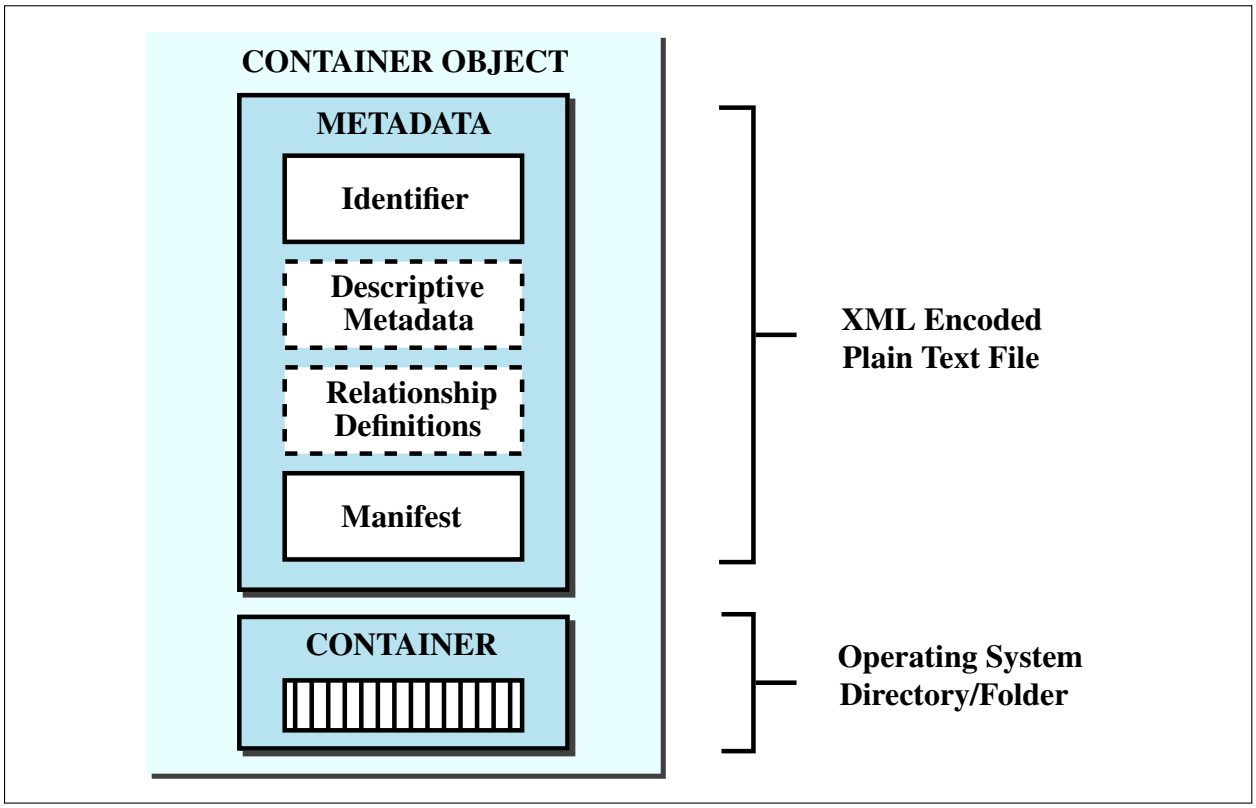


Figure 4-3. Simple repository container object component structure

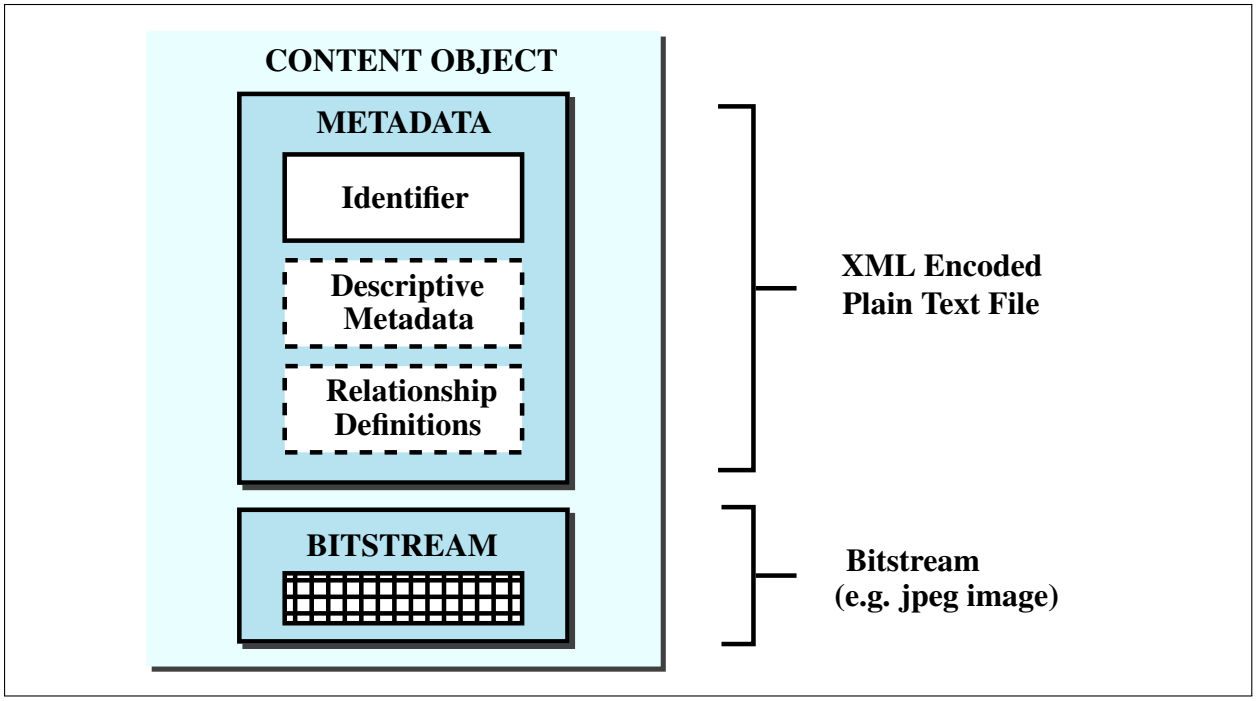


Figure 4-4. Simple repository digital object component structure

Chapter 5

Case studies

In order to assess the overall effectiveness of the prototype simple repository design described in Chapter 4, repositories for two real-world case study collections were implemented. This chapter discusses the two case study implementations.

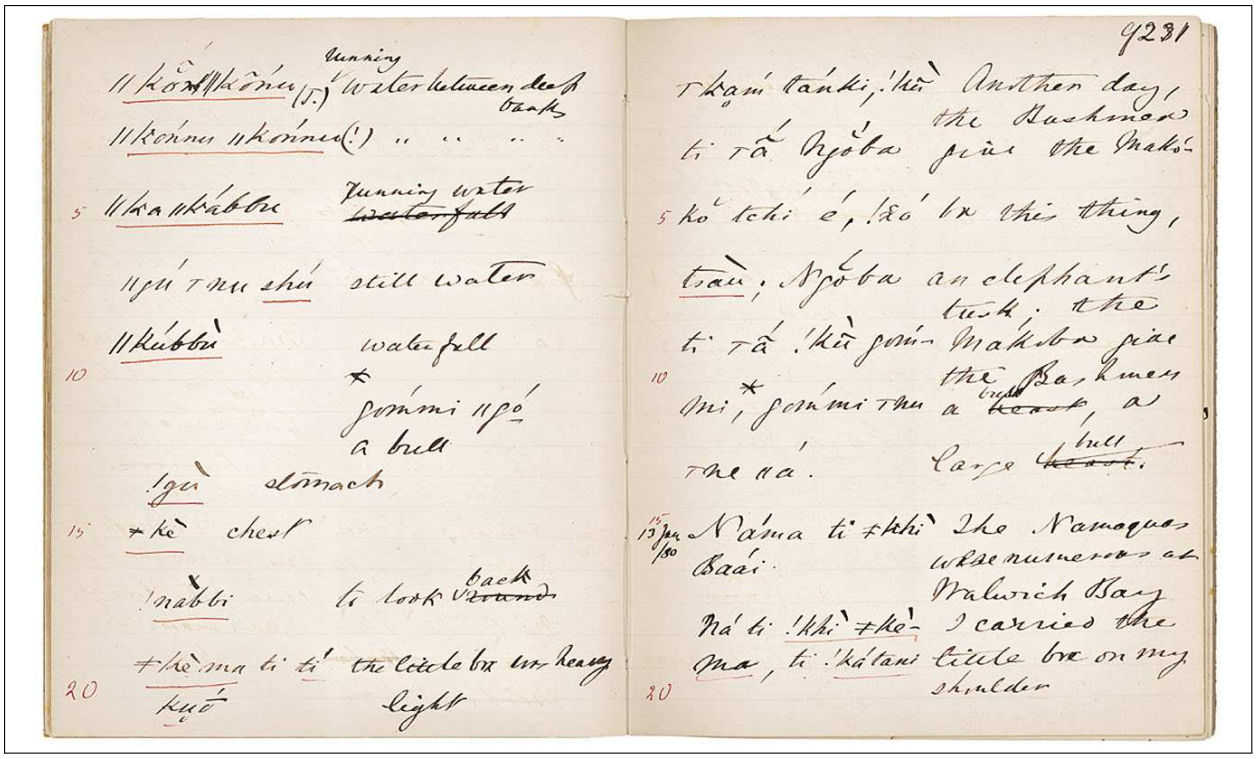


Figure 5-1. Screenshot showing a sample page from the “Posts and trading” story in the Lucy Lloyd !Kun notebooks

5.1 Bleek and Lloyd collection

5.1.1 Overview

The Bleek and Lloyd collection (Skotnes, 2007) is a 19th century compilation of notebooks and drawings comprising of linguistic and ethnographic work of Lucy Lloyd and Wilhelm Bleek on the life of the |Xam and !Kun Bushman people of Southern Africa. In 2003, the Lucy Lloyd Archive and Research centre at the University of Cape Town embarked on a large scale digitisation project and all the artifacts are in the process of being scanned and corresponding representation information generated. Table 5-1 shows the current composition of the digitised objects and Figure 5-1 shows a sample page from one of the digitised notebooks.

Table 5-1. Bleek& Lloyd collection profile

Collection theme	Historical artifacts; museum objects
Media types	Digitised
Collection size	6.2GB
Content type	image/jpeg
Number of collections	6
Number of objects	18 924

5.1.2 Object storage

Table 5-2. Bleek& Lloyd repository item classification

Item	Object Type	Comments
Notebook	Container object	Author compilation of books
Book	Container object	Compilation of digitised pages
Story	Content object	Content object without bitsreams
Page	Content object	Digitised page

Table 5-2 shows the object composition of the collection and Figure 5-2 shows the relationships among the objects.

The metadata objects are encoded using Dublin Core (*Dublin Core Metadata Element Set, Version 1.1* 1999); Listings 5-1, 5-2 and 5-3 show sample encoding for Content Objects, “virtual” Content Objects and Container Objects.

Listing 5-1. A digital content metadata file

```
<?xml version="1.0" encoding="utf-8"?>
<resource xmlns:dcterms="http://purl.org/dc/terms/">
  <dcterms:requires>...</dcterms:requires>
</resource>
```

Listing 5-2. A virtual object metadata file

```
<?xml version="1.0" encoding="utf-8"?>
<resource xmlns:bl="http://lloydbleekcollection.uct.ac.za/"
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:dcterms="http://purl.org/dc/terms/">
  <dc:identifier>2</dc:identifier>
  <dc:title>Words and sentences</dc:title>
  <dcterms:contributor>Adam Kleinhardt</dcterms:contributor>
  <dc:subject>Words and sentences</dc:subject>
  <bl:keywords>
   ...
  </bl:keywords>
  <dc:description>...</dc:description>
  <bl:comments>...</bl:comments>
  <dcterms:created>July 1866</dcterms:created>
  <bl:pages>001-066</bl:pages>
  <dcterms:requires>A1_4_1_00001.JPG</dcterms:requires>
   ...
  <dcterms:requires>A1_4_1_00066.JPG</dcterms:requires>
</resource>
```

Listing 5-3. A container object metadata file

```
<?xml version="1.0" encoding="utf-8"?>
<resource xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:dcterms="http://purl.org/dc/terms/">
  <dc:title>BC_151_A1_4_001</dc:title>
  <dcterms:hasPart>A1_4_1_FUCOV.JPG</dcterms:hasPart>
  <dcterms:hasPart>A1_4_1_IFCOV.JPG</dcterms:hasPart>
  <dcterms:hasPart>A1_4_1_00001.JPG</dcterms:hasPart>
   ...
  <dcterms:hasPart>A1_4_1_INS45.JPG</dcterms:hasPart>
  <dcterms:hasPart>A1_4_1_IBCOV.JPG</dcterms:hasPart>
  <dcterms:hasPart>A1_4_1_BUCOV.JPG</dcterms:hasPart>
  <dcterms:hasPart>A1_4_1_SPINE.JPG</dcterms:hasPart>
</resource>
```

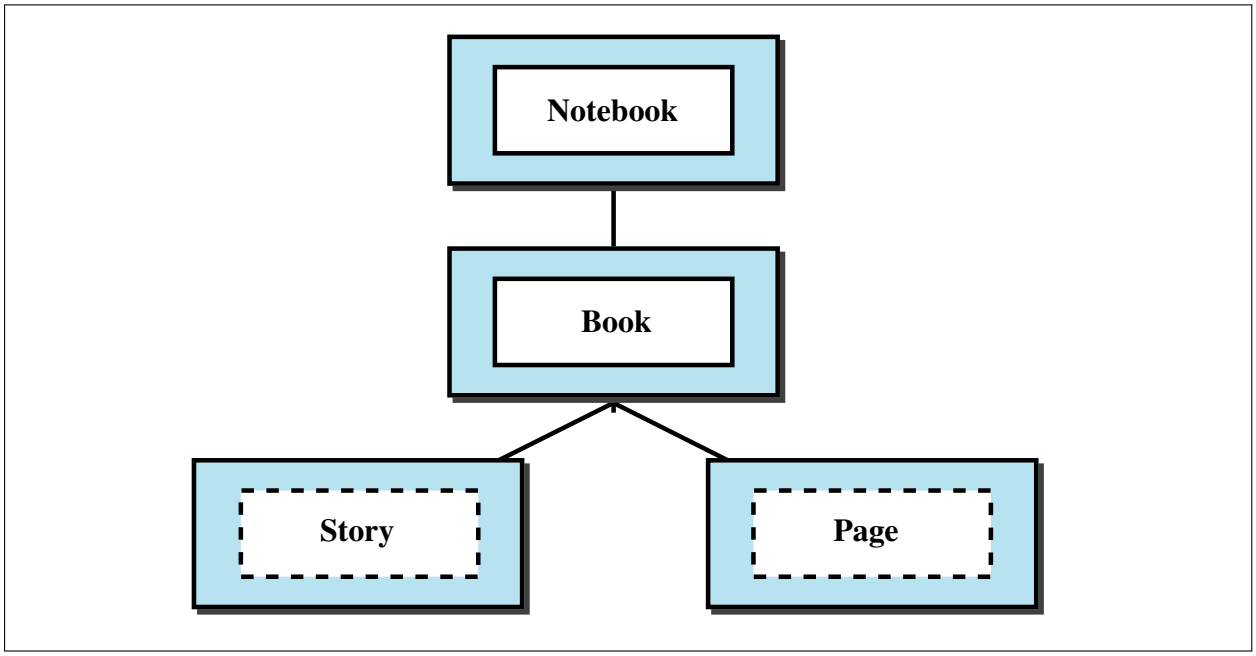


Figure 5-2. Collection digital object component structure

5.1.3 Digital Library Systems

The Digital Bleek and Lloyd collection

The digital Bleek and Lloyd collection (*The Digital Bleek and Lloyd* 2007) is an online[1] catalogue that was developed to store and enable access to digitised manuscripts described in Section 5.1.1. The underlying software was initially designed to enable access to as many people as possible so usage requirements were minimal—it was not even necessary to use a Web server or database. The system was designed to be XML-centric, and is based on an implementation strategy that involves pre-generating scalable hyperlinked XHTML pages using XSLT (Suleman, 2007). However, the original system was not focused on preservation, extensibility or reusability. In an attempt to take advantage of these attributes and also simplify the resulting system, a prototype redesigned system (Phiri and Suleman, 2012) was developed using the repository described in Section 5.1.2 as the underlying data storage layer.

Bonolo

The Bonolo project—undertaken in 2012—was initiated to investigate a new approach to building digital repository systems (Hammer and Robinson, 2011). One of the project deliverable is a prototype generic DLS (Hammer and Robinson, 2011; Phiri et al., 2012) that makes use of the repository described in Section 5.1.2 as the data storage layer.

[1] http://lloydbleekcollection.cs.uct.ac.za

Figure 5-3. Screenshot showing the Die Mond South plant fossil from the Eastern Cederberg rock art site

5.2 SARU archaeological database

5.2.1 Overview

The Department of Archaeology[2]'s Spatial Archaeology Research Unit (SARU) at the University of Cape Town has been compiling archaeological collections since the early 1950s. These collections are predominantly in the form of site records and corresponding artifacts within the vicinity of the sites. Table 5-3 show the composition of collections that have been compiled thus far, and Figure 5-3 shows an image of a rock art motif from one of the archaeological sites.

Owing to the growing number of collections and a growing need by a number of researchers to access this information, an archaeological database was designed in 2005, in part, to produce layers suitable for integration with Geographic Information Systems. The site records are currently accessed via a Microsoft Access[3] database-based desktop application used to store the digital archive (Wiltshire, 2011).

[2] http://web.uct.ac.za/depts/age
[3] http://office.microsoft.com/en-us/access

Listing 5-4. A sample kloof/farm container object metadata file

```
<?xml version="1.0" encoding="utf-8"?>
<resource xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:dcterms="http://purl.org/dc/terms/">
  <dc:title>Posen</dc:title>
  <dcterms:hasPart>POS1</dcterms:hasPart>
  <dcterms:hasPart>POS2</dcterms:hasPart>
  ...
  <dcterms:hasPart>POS11</dcterms:hasPart>
  <dcterms:hasPart>POS12</dcterms:hasPart>
</resource>
```

Table 5-3. SARU archaeological database collection profile

Collection theme	Archaeology artifacts; museum objects
Media types	Born digital
Collection size	283GB
Content type	image/jpeg; image/tiff
Number of collections	110
Number of objects	72 333

5.2.2 Object storage

The records from the database were re-organised to conform to the design described in Chapter 4. Table 5-4 shows the object types identified in the collection and Figure 5-4 is an illustration of the repository structure and relationships among the objects.

Table 5-4. SARU repository item classification

Item	Object Type	Comments
Map Sheet	Container Object	Map sheet code
Farm/Kloof	Container Object	Farm/Kloof
Site Number	Container Object	Site number
Project/Recorder	Container object	Project, recorder or contributor
Artifact	Content object	Photograph

The metadata records were encoded using a custom tailored metadata scheme, conforming to the original format of data input forms used by research when conducting field studies. Listings 5-4 and 5-5 show encoding for a sample site record Container object and Content object, respectively.

Listing 5-5. A sample site record content object metadata file

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
  <recordId>2809</recordId>
  <siteNo>POS12</siteNo>
  <mapSheet>
    ...
  </mapSheet>
  <localName>Posen 12</localName>
  <commonName>NULL</commonName>
  <project>
    ...
  </project>
  <recorder>
    ...
  </recorder>
  <date>2007-04-02 00:00:00</date>
  <directionsToSite>NULL</directionsToSite>
  <plottedOnMap>0</plottedOnMap>
  <commentsOnSite>NULL</commentsOnSite>
  <width>NULL</width>
  <depth>NULL</depth>
  <length>NULL</length>
  <breadth>NULL</breadth>
  <previousRecordings>NULL</previousRecordings>
  <gpsLatitude>-34.04872</gpsLatitude>
  <gpsLongitude>22.27378</gpsLongitude>
  <altitude>NULL</altitude>
  <time>NULL</time>
  <grading>NULL</grading>
  <f12>NULL</f12>
  <siteType>
    ...
  </siteType>
  <category>
    ...
  </category>
  <description>
    ...
  </description>
  <contents>
    ...
  </contents>
</site>
```

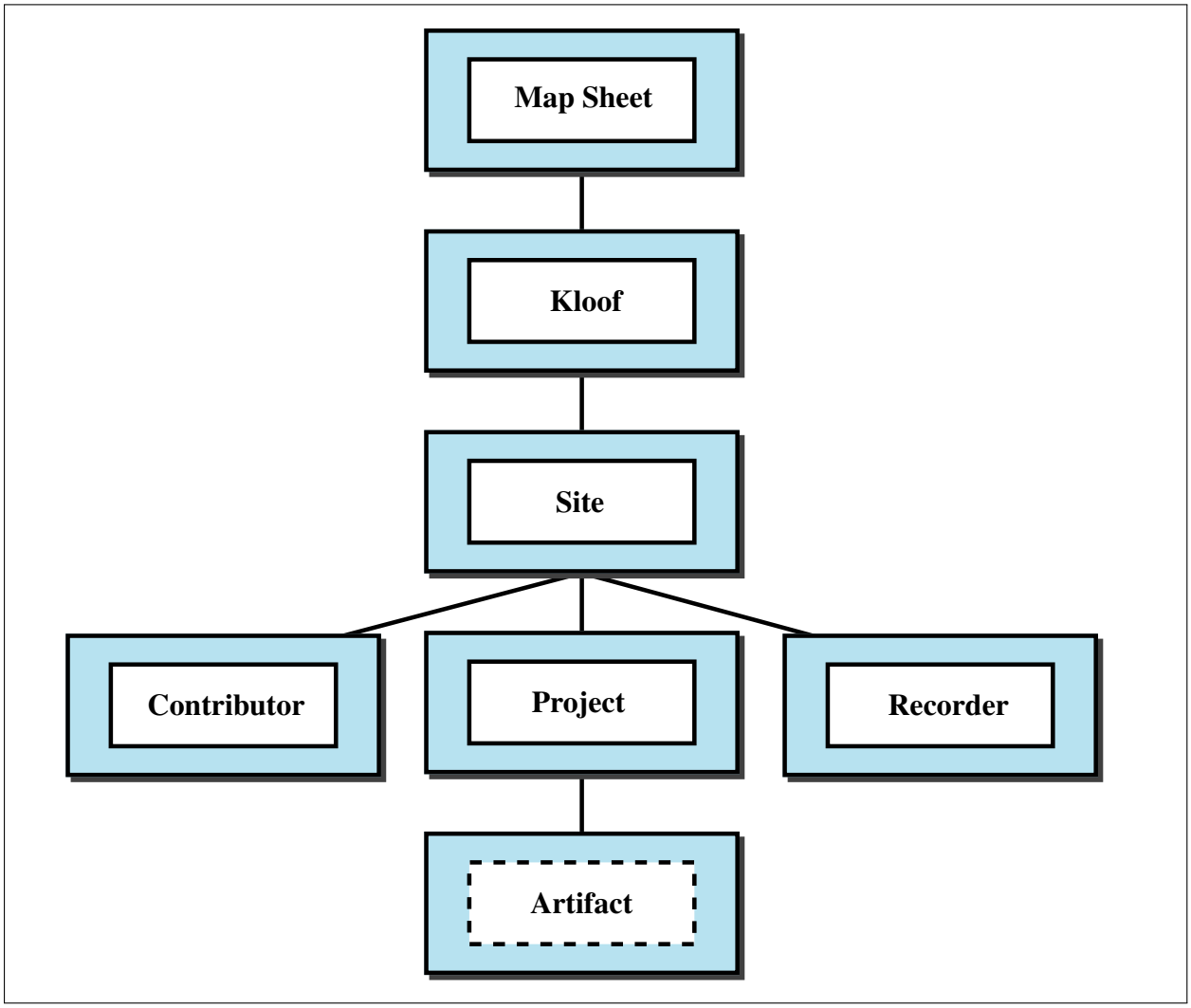


Figure 5-4. Collection digital object component structure

5.2.3 Digital Library Systems

School of rock art

The School of Rock Art (Crawford, Lawrence, and Marston, 2012) is a Web application that was developed to act as an archaeology educational tool for elementary school students. The Web application is composed of three independent modules that all interact with the repository described in this section.

5.3 Summary

This chapter presented two real-world case study collections based on the simple prototype repository design outlined in Chapter 4. Furthermore, some DLS implementations that used each of the case study collections as repository sub-layers were highlighted.

In essence, the case studies serve as proof of concept implementations for real-world practical application of the simple design. In Chapter 6, Section 6.1, a user study, in the form of a developer-oriented survey that used the Bleek and Lloyd collection as the primary storage layer, is described.

Chapter 6

Evaluation

Evaluation of DLs has been a subject of interest for DL research from the very early stages. This is evidenced by early initiatives such as the D-Lib Working Group on DL Metrics (*D-Lib Working Group on Digital Library Metrics* 1998) that was established in the late 1990s. A series of related studies have since been conducted with the aim of outlining a systematic and viable way of evaluating the complex, multi-faceted nature of DLs that encompasses content, system and user-oriented aspects. For instance, the DELOS[1] Cluster on Evaluation (Borgman, Solvberg, and Kovács, 2002; *DELOS Workshop on the Evaluation of Digital Libraries* 2004), which is perhaps the most current and comprehensive DL evaluation initiative, was initiated with the aim of addressing the different aspects of DLs evaluation.

The DELOS DL evaluation activities have yielded some significant results; in an attempt to understand the broad view of DLs, Fuhr et al. (Fuhr et al., 2001) developed a classification and evaluation scheme using four major dimensions: data/collection, system/technology, users and usage, and further produced a MetaLibrary comprising of test-beds to be used in DL evaluation. In a follow-up paper, Fuhr et al. (Fuhr et al., 2007) proposed a new framework for evaluation of DLs with detailed guidelines for the evaluation process.

This research proposes simplifying the overall design of DLSes and more specifically designing for simplicity of management and ease of use the resulting DLSes. The design principles derived in Chapter 3 were used to design and implement a simple generic repository sub-layer for DLSes. In Chapter 5 three proof of concept file-based repository implementations are presented to evaluate the effectiveness of this approach.

A developer survey, outlined in Section 6.1, was conducted to assess the impact of redesigning the repository sub-layer on extensibility of implementations based on this design.

Furthermore, owing to the fact that repositories have a potential to grow, detailed scalability performance benchmarks were conducted to assess the performance of this design strategy relative to the sizes of collections; these performance benchmarks are outlined in Section 6.2.

[1] http://www.delos.info

6.1 Developer survey

The developer-oriented user study was conducted to assess the simplicity of file-based repository implementations and the easy of interaction with such implementations.

6.1.1 Target population

The survey participants were recruited from a total of 34 Computer Science Honours (CSC4000) students, enrolled for the World Wide Web Technologies (WWW) elective course module at the University of Cape Town.

The WWW module had a mandatory practical assignment, accounting for 20 % of the overall assessment, in which the students were required to build generic Web applications, in groups, using the file-based repository store described in Section 5.1. Screenshots of the online questionnaire are in the Appendix section[2], and show the assignment question. A request for survey participation was emailed to the class mailing list after the assignment due date, in which 26 out of the 34 students responded, as shown in Table 6-1

Table 6-1. Developer survey target population

	Group 1	**Group 2**	**Group 3**	**Group 4**	**Group 5**	**Group 6**	**Group 7**	**Group 8**	**Group 9**	**Group 10**	**Group 11**	**Group 12**
Candidates	3	3	3	3	3	2	3	3	3	3	3	2
Respondents	3	3	1	2	2	3	3	1	2	2	1	2

6.1.2 Data collection

A post-experiment survey was conducted in the form of an online questionnaire[3], designed using LimeSurvey[4]. The questionnaire was aimed at eliciting participants' experience in working with a file-based collection.

6.1.3 Results

The survey participants' background-related information is shown in Figures 6-1, 6-2 and 6-3. The implementation of the Web services was done using a variety of programming languages, as shown in Figure 6-4.

[2]Please see Appendix A.2 for details
[3]Please see to Appendix A.2 for details
[4]http://www.limesurvey.org

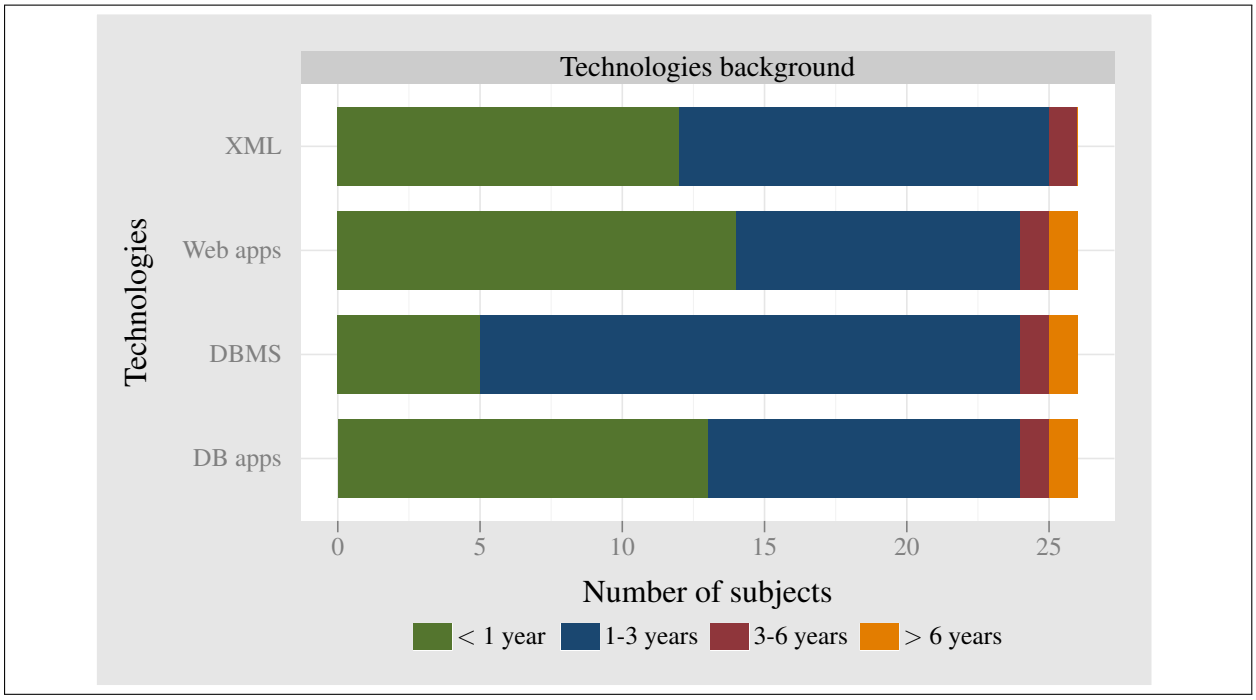


Figure 6-1. Survey participants' background knowledge working with technologies relevant to the study.

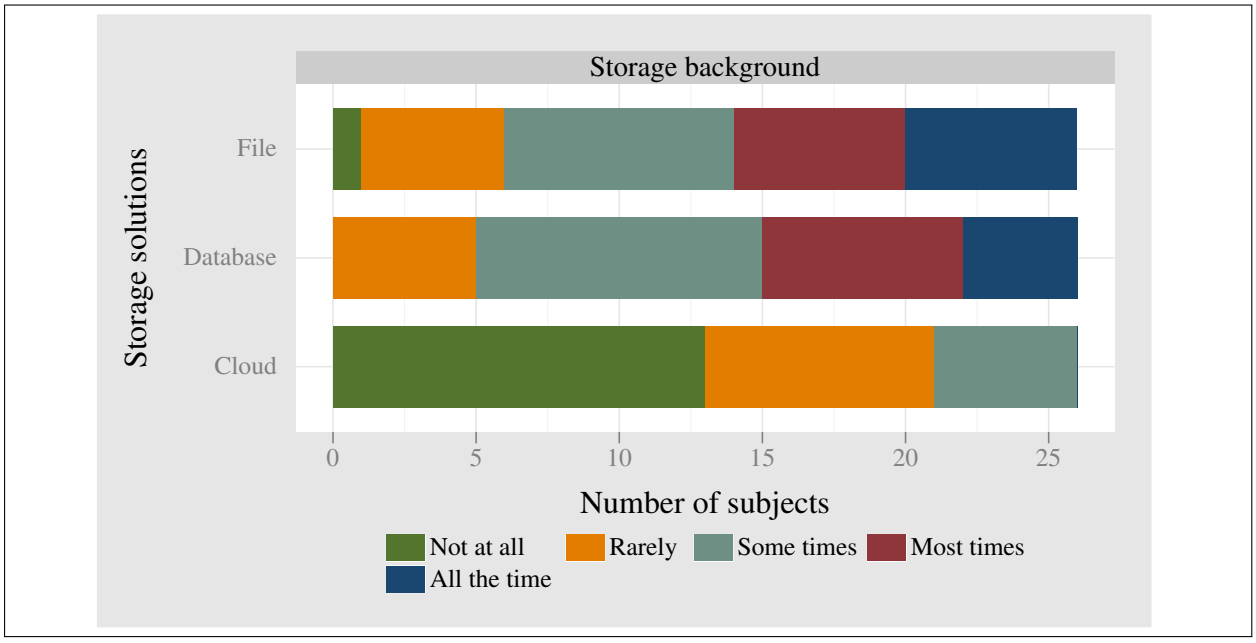


Figure 6-2. Survey participants' background working with some selected popular storage solutions.

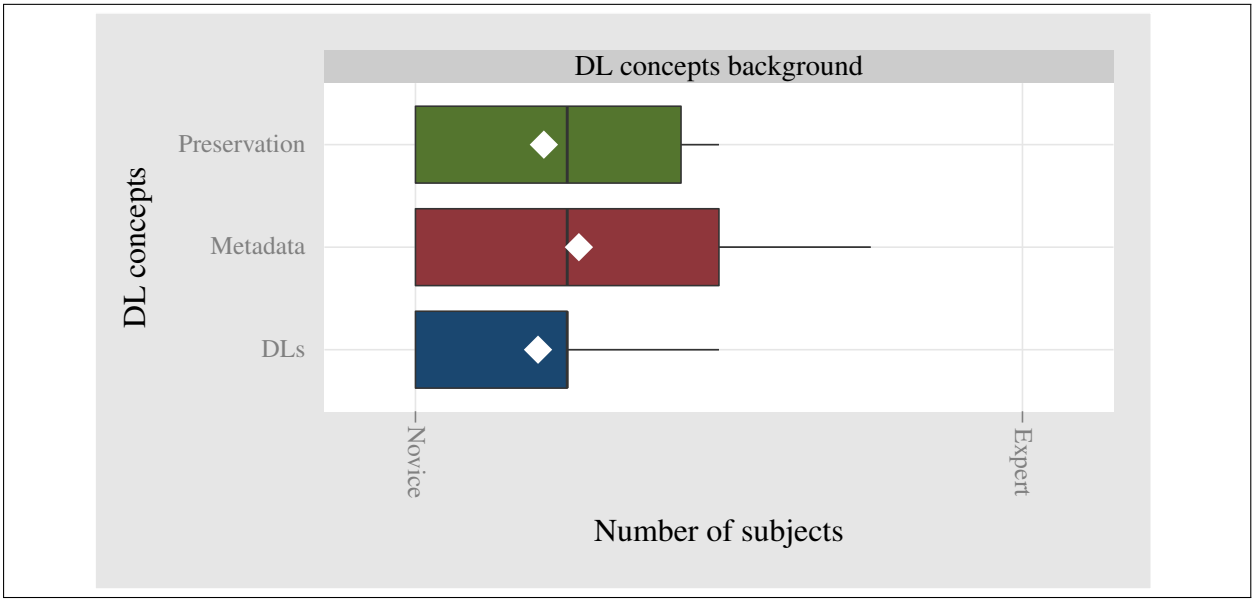


Figure 6-3. Survey participants' knowledge of some fundamental DL concepts.

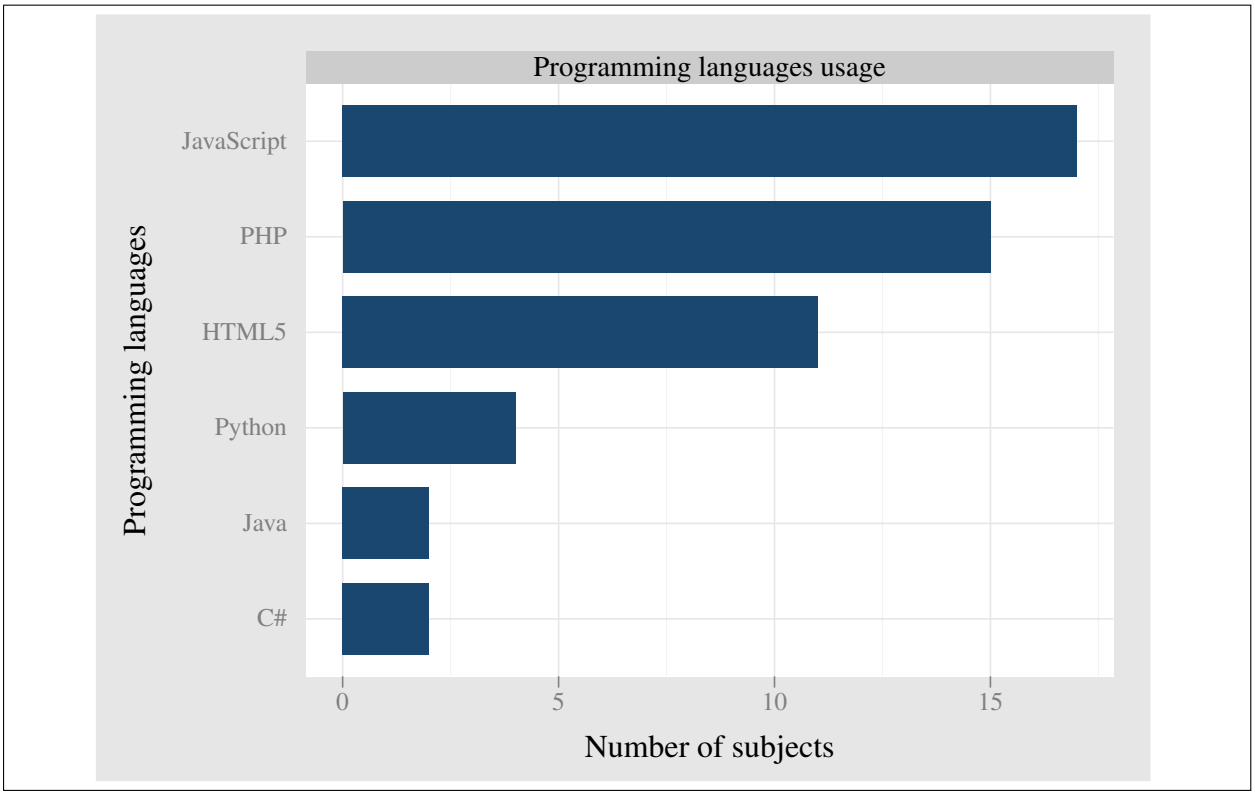


Figure 6-4. Survey participants programming languages usage during service implementation.

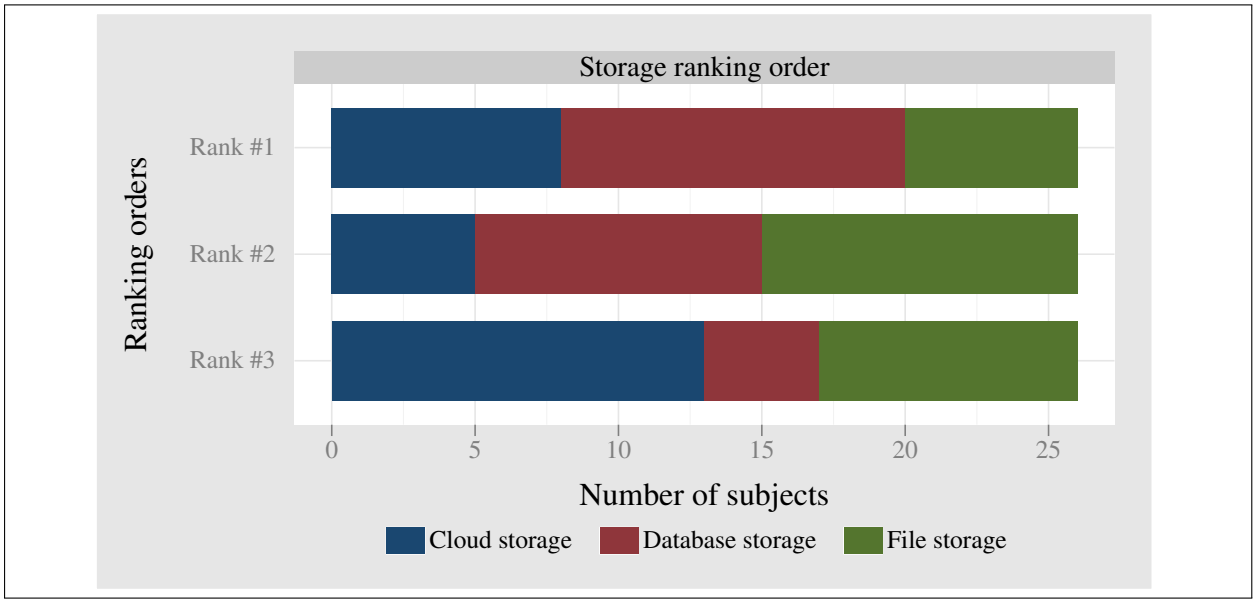


Figure 6-5. Survey participants' rankings of data storage solutions.

The respondents' views on the simplicity and ease of use of the repository is shown in 6-6; additionally, their rankings of possible storage solutions for metadata records are shown in Figure 6-5. Finally, their preferences on the possible solutions to use for data management tasks/operations are shown in Figure 6-7.

6.1.4 Discussion

The survey results indicate that the target population generally had the necessary skillset required for this study. The majority of respondents had some form of experience working with Web applications and associated technologies (see Figure 6-1); the majority of them frequently worked with Database Management Systems and had some form of experience working with file-based systems (see Figure 6-2). In addition, all respondents were familiar with fundamental concepts associated with DLs (see Figure 6-3).

The range of Web services implemented by the target population and the variety of programming languages used to implement the services is indicative of the flexibility of the repository design. Furthermore, these results strongly suggest that the repository design did not significantly influence the choice of service and implementation language. This conclusion is further supported by an explicit survey question in Figure A-5, which was aimed at eliciting respondents' views on whether the repository structure had a direct influence on their programming language(s) of choice, to which 15 % of the participants agreed.

The strong preference of using databases as storage structures, shown in the results from Figures 6-5 and 6-7 is arguably as a result of the majority of participants' prior work with databases, and is best explained by the question that asked participants for reasons prior for their preferred storage solutions; the responses from some participants who ranked databases first are listed below.

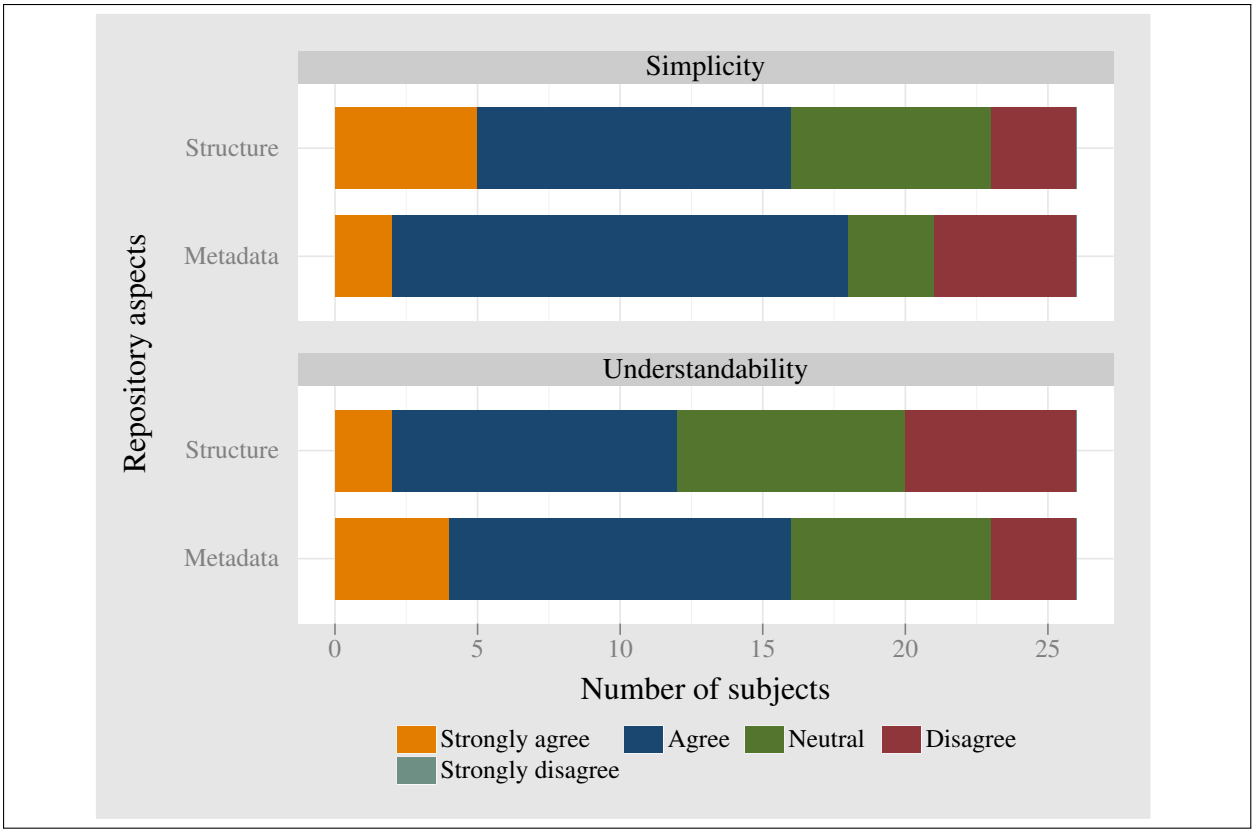


Figure 6-6. Survey participants' simplicity and understandability ratings of repository design.

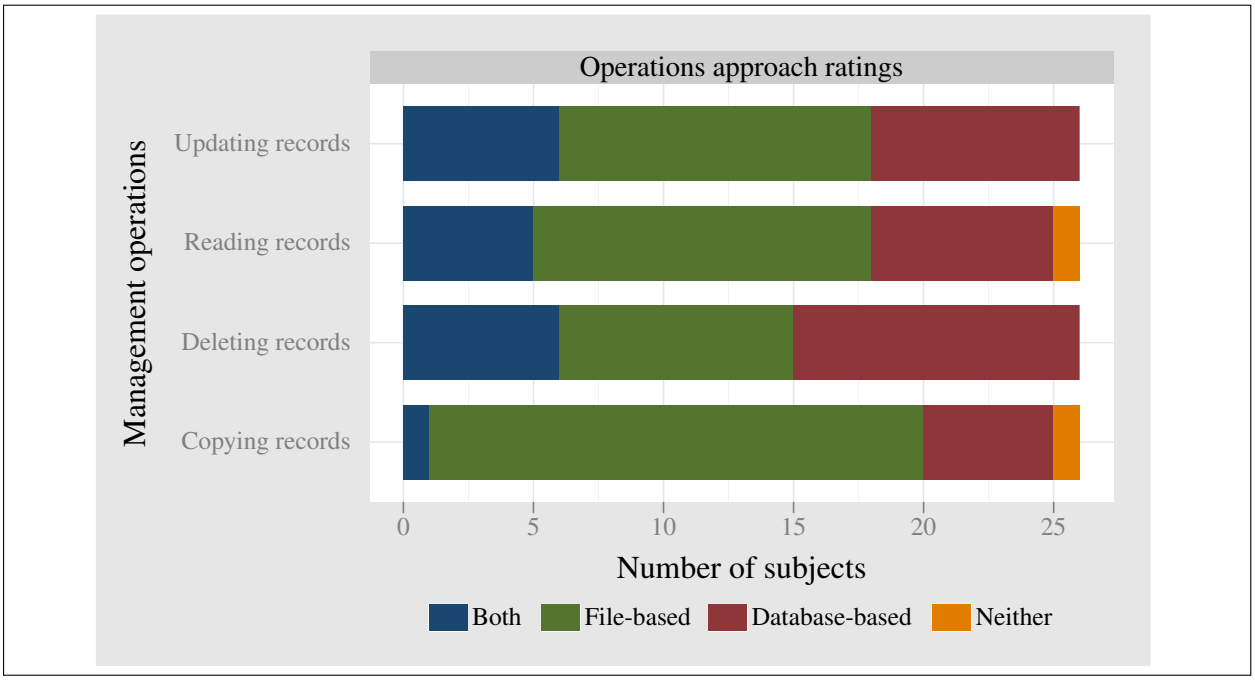


Figure 6-7. Survey participants' ratings of data management approaches for DL operations.

- "I understand databases better."
- "Simple to set up and sheer control"
- "Easy setup and connection to MySQL database"
- "Speed of accessing data, and its free."
- "Ease of data manipulation and relations"
- "Easy to query"
- "Centralised management, ease of design, availability of support/literature"
- "The existing infrastructure for storing and retrieving data"
- "Querying a database table to retrieve a record is most useful for data."

Interestingly, out of the total 12 participants whose preference was databases,the majority identified themselves as having little background information pertaining to metadata standards, DLs and digital preservation. It can be argued that their lack of knowledge of these fundamental concepts could have influenced their subjective views. This is supported by some of their general comments listed below.

- "Had some difficulty working the metadata, despite looking at how to process DC metadata online, it slowed us down considerably."
- "Good structure although confusing that each page has no metadata of its own(only the story)."
- "The hierarchy was not intuitive therefore took a while to understand however having crossed that hurdle was fairly easy to process."
- "I guess it was OK but took some getting used to"

6.1.5 Summary

The results from the developer survey showed that developer interaction with resulting systems is not significantly affected. More importantly, the results indicate that DLS management tasks could potentially be simplified.

6.2 Performance

A significant architectural change performed to the design and implementation of the simple repository outlined in Chapter 4 involves changing the way metadata records are stored in the repository sub-layer of DLSes. More specifically, the proposed solution advocates for the use of a typical operating system file system for the storage of metadata records, as opposed to the conventional use of a database management system. This design decision is motivated by two key factors—simplicity and manageability. However, conventional wisdom (Nicola and John, 2003;

Sears, Ingen, and Gray, 2007) points to the fact that system performance would evidently be adversely affected for relatively large collections.

The remainder of this section outlines the performance experiments conducted to evaluate the simple repository design. Section 6.2.1 briefly describes the test environment set-up to conduct the experiments; Section 6.2.2 describes the test dataset and Secton 6.2.3 describes the workloads used during experimentation. In Section 6.2.4 a series of performance benchmarks are discussed, and a discussion of performance comparisons with DSpace is then discussed in Section 6.2.5.

6.2.1 Test setup

The experiments were all conducted locally—to isolate network-related hidden factors that could distort the measurements—on a standalone Intel Pentium (E5200@ 2.50 GHz) with 4 GB of RAM running Ubuntu 12.04.1 LTS. Apache 2.2.22 Web server and Jetty were used to host module implementations; and ApacheBench 2.3 and Siege 2.70 were used to simulate a single user request, with five run-averages taken for each aspect request.

Furthermore, in order to isolated computing resource hidden factors such as memory and CPU usage, the only applications that were set-up and subsequently executed on the machine were those related to the experiments being conducted. Table 6-2 shows a summary of the configurations that were used to conduct the experiments.

Table 6-2. Performance experiment hardware and software configuration

Hardware	Pentium(R) Dual-Core CPU E5200@ 2.50 GHz 4 GB RAM
Software	Apache/2.2.22 (*The Apache HTTP Server Project* 2012) ApacheBench 2.3 (*Apache HTTP Server Version 2.2* 2012) Apache Solr 4.0 (*Apache Solr* 2012) Jetty 8.1.2 (*Jetty:// 2012*) Siege 2.70 (Fulmer, 2012) Ubuntu 12.04.1 LTS (*Ubuntu 12.04.2 LTS (Precise Pangolin)* 2012)

6.2.2 Test dataset

Table 6-3. Performance experiment dataset profile

Collection theme	Dublin Core encoded plain text files

(Continued on next page)

Table 6-3. (continued)

Collection size	8.6 GB
Content type	text/xml
Total number of objects	1 907 000

The dataset used for the experiments is a collection of XML records, encoded using simple Dublin Core, which were harvested from the NDLTD Union Catalog[5] using the OAI-PMH 2.0 protocol. Table 6-3 shows a summary of the dataset profile used for conducting the performance experiments, and the details of the repository and sub-collection structure are shown in Listing 6-1 and Listing 6-2 respectively.

The OAI-PMH unique *setSpec* element names, shown in Listing 6-2, for each of the harvested records were used to create container structures that represent collection names for the resulting archive.

6.2.3 Workloads

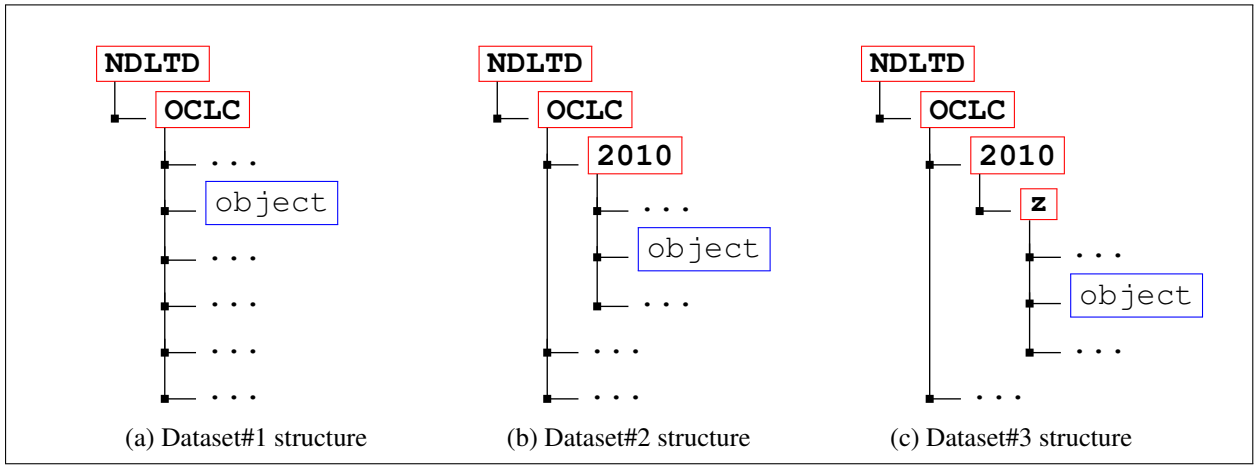
The 1 907 000 objects in the experiment dataset—summarised in Table 6-3—are aggregate metadata records from a total of 131 different institutional repositories from around the world; in addition the metadata records are encoded in Dublin Core, a metadata scheme which allows for all elements to be both optional and repeatable. As a result, the structure of the metadata records was not consistent throughout all the records. A random sampling technique was thus used to generate linearly increasing workloads, with records randomly selected from the 131 *setSpecs*.

Table 6-4 shows the 15 workloads initially modelled for use during the performance experimentation stage. An additional two datasets were then spawned to create experiment datasets with varying hierarchical structures. Table B-10 shows the profiles for the three dataset workload models, and Figure 6-8 illustrates the object organisation in one-level, two-level and three-level workload models.

Table 6-4. Experiment workload design for Dataset#1

	W1	**W2**	**W3**	**W4**	**W5**	**W6**	**W7**	**W8**	**W9**	**W10**	**W11**	**W12**	**W13**	**W14**	**W15**
Records	100	200	400	800	1600	3200	6400	12 800	25 600	51 200	102 400	204 800	409 600	819 200	1 638 400
Collections	19	25	42	57	67	83	100	112	116	119	127	129	128	131	131
Size [MB]	0.54	1.00	2.00	3.90	7.60	15.00	30.00	60.00	118.00	236.00	471.00	942.00	1945.00	3788.80	7680.00

[5] http://union.ndltd.org/OAI-PMH



(a) Dataset#1 structure (b) Dataset#2 structure (c) Dataset#3 structure

Figure 6-8. The workload hierarchical structures for the three experiment datasets. The $setSpec$, publication date and first character of creator name were used as first-, second- and third-level container names respectively.

Listing 6-1. NDLTD union catalog OAI-PMH Identity verb response

```
<?xml version="1.0" encoding="utf-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
         http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2012-09-23T10:19:23Z</responseDate>
  <request verb="Identify">http://union.ndltd.org/OAI-PMH/</request>
  <Identify>
    <repositoryName>NDLTD Union Archive of ETD Metadata</repositoryName>
    <baseURL>http://union.ndltd.org/OAI-PMH/</baseURL>
    <protocolVersion>2.0</protocolVersion>
    <adminEmail>hussein@cs.uct.ac.za</adminEmail>
    <earliestDatestamp>2011-09-07T02:15:34Z</earliestDatestamp>
    <deletedRecord>persistent</deletedRecord>
    <granularity>YYYY-MM-DDThh:mm:ssZ</granularity>
    <description>
      <eprints xmlns="http://www.openarchives.org/OAI/1.1/eprints"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.openarchives.org/OAI/1.1/eprints
        http://www.openarchives.org/OAI/1.1/eprints.xsd">
        <content>
          <URL>http://union.ndltd.org/</URL>
          <text>NDLTD Union Archive of ETD Metadata</text>
        </content>
        <metadataPolicy/>
        <dataPolicy/>
      </eprints>
    </description>
  </Identify>
</OAI-PMH>
```

Listing 6-2. NDLTD union catalog OAI-PMH ListSets verb response

```
<?xml version="1.0" encoding="utf-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
         http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2012-09-24T12:27:59Z</responseDate>
  <request verb="ListSets">http://union.ndltd.org/OAI-PMH/</request>
  <ListSets>
    <set>
      <setSpec>UPSALLA</setSpec>
      <setName>DiVA Archive at Upsalla University</setName>
    </set>
    <set>
      <setSpec>OCLC</setSpec>
      <setName>OCLC</setName>
    </set>
    <set>
      <setSpec>IBICT</setSpec>
      <setName>IBICT Brazilian ETDs</setName>
    </set>
    <set>
      <setSpec>VTETD</setSpec>
      <setName>Virgina Tech. Theses and Dissertation</setName>
    </set>
    ...
    ...
    ...
  </ListSets>
</OAI-PMH>
```

6.2.4 Benchmarks

A series of performance benchmarks were conducted on some typical DL services, in order to assess the overall performance of the architecture.

The purpose of the performance experiments was to evaluate the performance and scalability of collections as the workload—in relation to collection size—was increased. The performance experiments were carried out on the following list of services,—with the exception of indexing—derived from a transaction log analysis of a production digital library system[6]—a subject repository running EPrints 2.1.1.

- Item ingestion
- Full-text search
- Indexing operations
- OAI-PMH data provider
- Feed generation

The series of experiments were designed specifically to determine the break-even points at which performance and scalability drastically degrades. Nielsen's three important limits for response times (Nielsen, 1993) were used as a basis for determining desirable response times for varying workloads.

The detailed descriptions of the experiments conducted on the services/aspects now follows.

Item ingestion

The ingestion process for a typical DLS in part involves importation of metadata associated with the bitstreams being ingested. The purpose of experiments conducted for this aspect was to determine the relative ingestion performance of metadata records, in terms of response time, with varying workload sizes.

Experiment: Item ingestion response time This experiment was aimed at assessing the ingestion response time for the 15 workloads.

Methodology A single record was randomly harvested from the OCLC setSpec[7], using datestamp-based selective harvesting (Lagoze et al., 2002b), in order to harvest records that were created, deleted, or modified after the initial bulk harvesting described in Section 6.2.2. The second and third-level container objects were then created in advance, for workloads in which the container objects in question were not present, to isolate the latency that would result from creating missing containers. The ingestion process was then simulated through a script that read the

[6] http://pubs.cs.uct.ac.za

[7] The OCLC setSpec was common to all the 15 workloads

record to be ingested and wrote the contents of the record to each of the 15 workload collections. The times taken to successfully write the record to disk were then noted.

Results The experiment results are shown in Table 6-5 and Figure 6-9.

Table 6-5. Impact of structure on item ingestion performance

	Dataset #1			Dataset #2			Dataset #3		
	Ingestion[ms]	**Parsing**	**Disk write**	**Ingestion[ms]**	**Parsing**	**Disk write**	**Ingestion[ms]**	**Parsing**	**Disk write**
W1	4.79	63.14 %	36.86 %	4.15	96.40 %	3.60 %	4.12	96.67 %	3.33 %
W2	5.69	97.76 %	2.24 %	5.09	97.42 %	2.58 %	4.02	96.35 %	3.65 %
W3	2.79	95.69 %	4.31 %	2.87	95.53 %	4.47 %	2.94	95.51 %	4.49 %
W4	2.78	95.67 %	4.33 %	4.08	96.78 %	3.22 %	2.84	95.65 %	4.35 %
W5	2.84	95.42 %	4.58 %	2.86	95.57 %	4.43 %	2.94	95.64 %	4.36 %
W6	2.78	95.68 %	4.32 %	2.90	95.67 %	4.33 %	2.86	95.41 %	4.59 %
W7	3.33	96.10 %	3.90 %	2.81	95.76 %	4.24 %	2.89	94.96 %	5.04 %
W8	2.80	95.59 %	4.41 %	2.80	95.63 %	4.37 %	2.86	94.65 %	5.35 %
W9	2.80	95.71 %	4.29 %	2.86	95.31 %	4.69 %	4.95	56.82 %	43.18 %
W10	2.89	95.54 %	4.46 %	2.79	95.72 %	4.28 %	2.88	95.52 %	4.48 %
W11	2.96	95.33 %	4.67 %	2.81	95.33 %	4.67 %	2.95	95.45 %	4.55 %
W12	3.95	96.26 %	3.74 %	2.96	95.40 %	4.60 %	2.87	95.62 %	4.38 %
W13	2.92	95.27 %	4.73 %	3.13	95.96 %	4.04 %	2.81	95.69 %	4.31 %
W14	2.83	95.45 %	4.55 %	2.85	95.63 %	4.37 %	2.78	95.66 %	4.34 %
W15	2.93	95.38 %	4.62 %	2.95	95.48 %	4.52 %	2.82	95.50 %	4.50 %

Discussion The ingestion response times remain constant irrespective of the workload size. This is because the only overhead incurred results from disk write IO. It should be noted that this experiment mimicked an ideal situation where the destination location for the item is known before hand.

The workload size does not affect the ingestion response time.

Full-text search

The purpose of these experiments was to determine the impact on collection size on query performance for indexed and non-indexed collections.

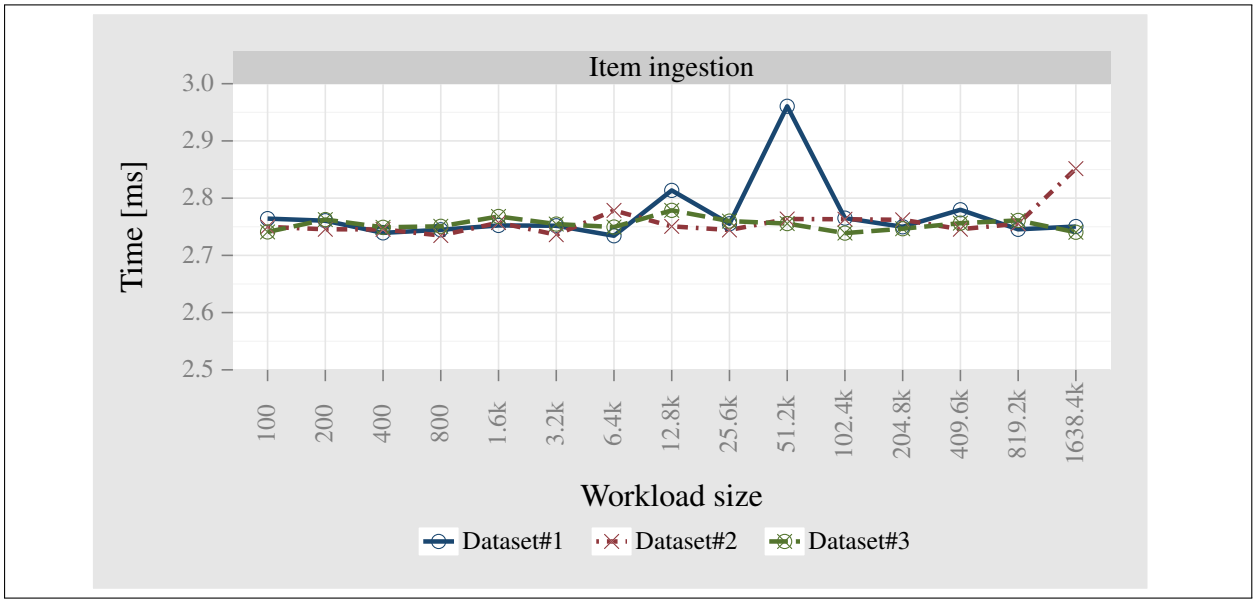


Figure 6-9. The average time, in milliseconds, taken to ingest a single item into an existing collection.

Experiment: Search performance for unindexed collections This experiment was conducted to determine query performance of non-indexed collections.

Methodology The most frequently occurring terms in the workloads were identified and search requests issued to determine response times. The search module implementation involved traversing collection containers and successively parsing and querying each metadata file in the collection for the search phrase in question.

Results The mean response times taken to generate search query resultsets are shown in Figure 6-10. In order to ascertain the overall distribution of the search response times, the time taken for the various search phases—directory traversal, parsing and XPath querying—was noted; Table 6-6 and Figure 6-10 show these times for the 15 workloads.

Table 6-6. Baseline performance benchmarks for full-text search

	Time [ms]	Traversal	Parsing	XPath
W1	24.67	16.13 %	38.49 %	45.38 %
W2	47.87	15.40 %	39.05 %	45.54 %
W3	97.38	14.89 %	39.01 %	46.10 %
W4	191.90	14.28 %	39.45 %	46.27 %
W5	386.99	13.82 %	39.39 %	46.79 %
W6	768.35	13.58 %	39.58 %	46.84 %
W7	1531.06	13.55 %	39.37 %	47.08 %
W8	3093.12	13.41 %	39.52 %	47.08 %

(Continued on next page)

Table 6-6. (continued)

	Time [ms]	Traversal	Parsing	XPath
W9	6172.22	13.37 %	39.76 %	46.87 %
W10	12 487.06	13.66 %	39.72 %	46.63 %
W11	25 108.74	14.20 %	39.49 %	46.31 %
W12	49 301.45	13.51 %	39.65 %	46.85 %
W13	100 267.33	14.20 %	39.85 %	45.95 %
W14	7 365 254.00	1.99 %	95.57 %	2.44 %
W15	18 664 713.65	5.28 %	92.87 %	1.85 %

Discussion The results in Figure 6-10 indicate an increasing linear correlation between the workload size and the query response time. This is largely due to the fact that all metadata records need to be analysed each time a search query is issued.

In addition, Table 6-6 indicates that a significant amount of time is spent parsing and querying the records, with each of the tasks accounting for an average of 39 % and 46 % respectively. Furthermore, this occurs before the workload size exceeds 409 600, at which point the parsing phase becomes extremely expensive—accounting for 95 % of the total search query time.

The query response time increases linearly as the workload size is increased and is drastically affected by larger workloads. The only effective way to get better performance would be to use an index.

Experiment: Impact of collection structure on search performance This experiment was conducted to assess the search query response times relative to a collection structure. The results obtained in Section 6.2.4, derived from a one-level collection structure, were compared with workloads of varying levels.

Methodology The search queries issued in Section 6.2.4 were issued to two-level and a three-level, illustrated in Figure 6-8b and Figure 6-8c respectively, structured workloads. The response times were noted and compared with those obtained from one-level structured workloads.

Results Table 6-7 shows the change in response times for two-level and three-level workloads relative to one-level workloads; and Figure 6-11 is a graphical representation of the response times for the different search query phases.

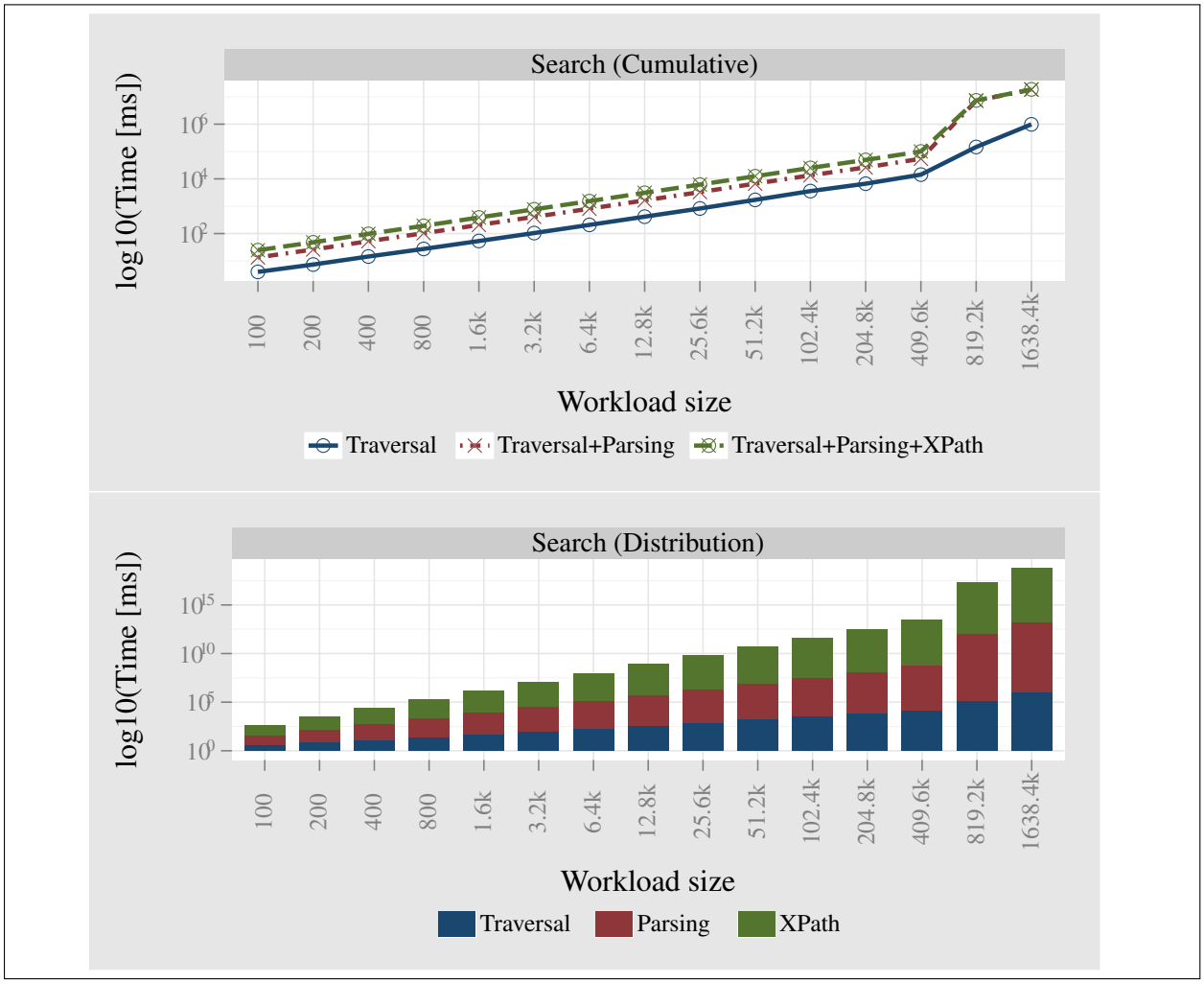


Figure 6-10. The cumulative times taken for the different search query processing phases—directory traversal, XML parsing and XPath query times.

Table 6-7. Search query time change relative to baseline

	Δ Dataset#2			Δ Dataset#3		
	Traversal	**Parsing**	**XPath**	**Traversal**	**Parsing**	**XPath**
W1	62.91 %	0.21 %	0.33 %	141.59 %	(0.39) %	2.13 %
W2	55.90 %	(0.99) %	0.45 %	134.24 %	(1.98) %	0.63 %
W3	51.78 %	0.65 %	1.65 %	126.18 %	(0.84) %	1.68 %
W4	39.22 %	0.38 %	1.24 %	113.29 %	(1.03) %	2.49 %
W5	32.61 %	0.82 %	0.36 %	101.85 %	(1.16) %	0.90 %
W6	22.81 %	0.14 %	0.76 %	83.80 %	(1.08) %	2.01 %
W7	16.72 %	1.68 %	(0.24) %	133.23 %	(0.03) %	0.46 %
W8	11.03 %	0.40 %	(0.72) %	59.56 %	(0.33) %	(0.02) %
W9	6.82 %	0.10 %	(0.27) %	48.84 %	(0.55) %	0.39 %
W10	7.47 %	(0.88) %	(0.89) %	42.11 %	(0.42) %	0.18 %
W11	0.18 %	(0.51) %	(0.70) %	38.81 %	0.44 %	(0.34) %

(Continued on next page)

Table 6-7. (continued)

	Δ Dataset#2			Δ Dataset#3		
	Traversal	**Parsing**	**XPath**	**Traversal**	**Parsing**	**XPath**
W12	9.39 %	1.07 %	(0.10) %	62.04 %	2.54 %	0.79 %
W13	8.61 %	0.01 %	0.54 %	137.16 %	1.84 %	2.72 %
W14	(30.37) %	(54.99) %	5.10 %	66.06 %	(86.83) %	6.78 %
W15	(79.22) %	(39.63) %	1.78 %	(61.31) %	(86.39) %	11.56 %

Discussion There is a significant linear increase in the search query response times before the workload size goes beyond 409 600, with the Parsing and XPath times remaining constant as the traversal times change.

Indexing operations

The integration of DLSes with indexing services is increasingly becoming common to facilitate seamless discovery of information through search and browse services. The experiments conducted for the index evaluation aspect were aimed at benchmarking various indexing operations associated with digital collections.

The Apache Solr search platform was deployed within a Jetty Servlet engine, and subsequently integrated with the 15 workloads. The workloads were conveniently set-up as 15 separate Apache Solr Cores and the following factors were investigated relative to the different workload sizes described in Section 6.2.3.

- Batch indexing of collections
- Incremental updates to existing collections

Experiment: Batch collection indexing benchmarks Batch indexing of collections is a common use-case; for instance, storage and retrieval systems will in certain instances require re-indexing of content when records have been updated with new values. This experiment was aimed at benchmarking the batch indexing of varying workloads.

Methodology The Apache Solr Data Import Request Handler (*Solr Wiki* 2012a) was configured for all the 15 workload cores to perform full builds. A full data-import command was then issued, and repeated 5 times, for each of the 15 workload cores. The minimum time taken to perform successful data-import operations was then recorded.

Results The batch indexing experiment results are shown in Table 6-8 and Figure 6-12.

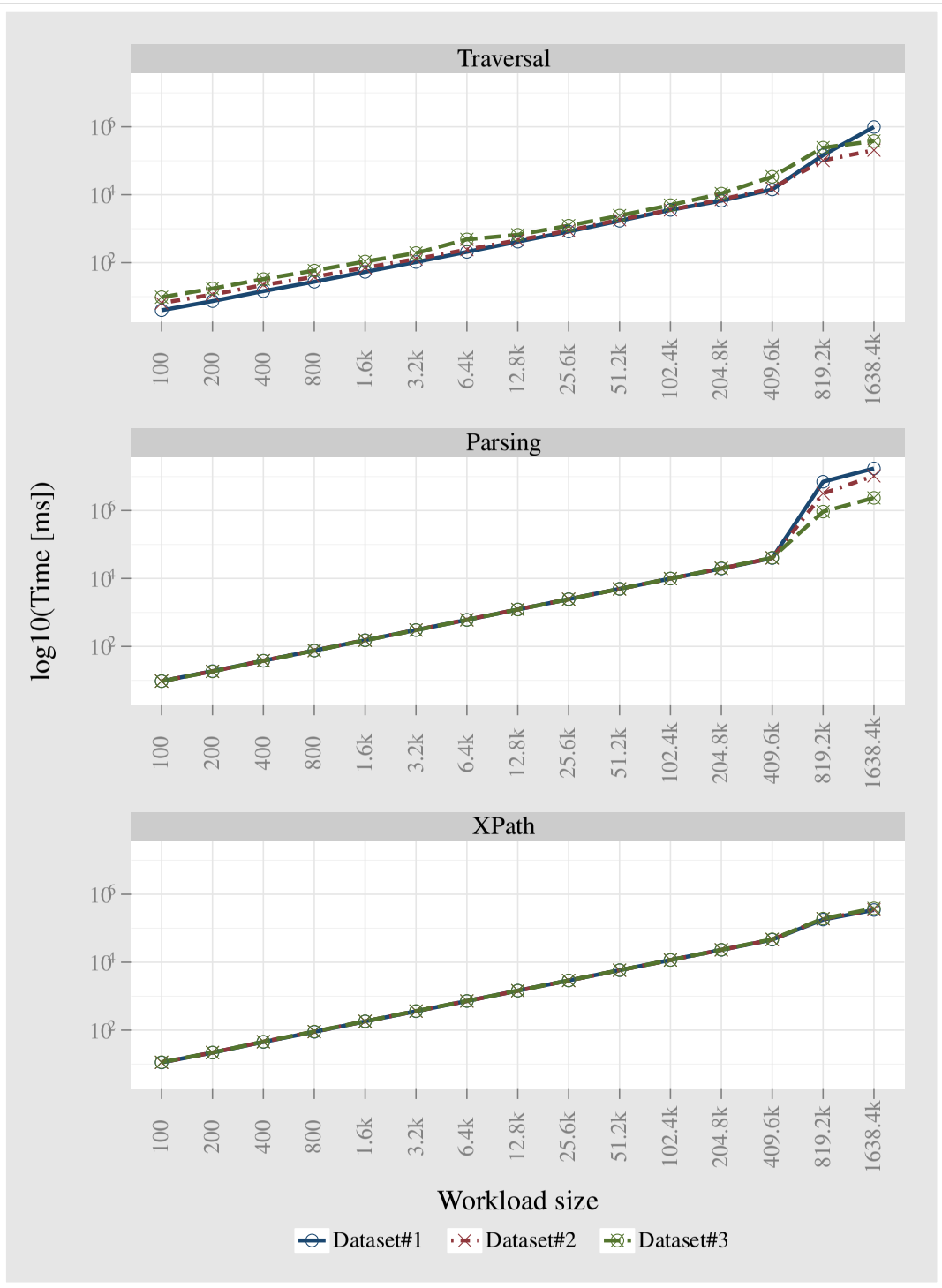


Figure 6-11. Impact of directory structure on query performance, split up into the different search phases—traversal, parsing and xpath phases.

Discussion The results strongly indicate that there is a linear relationship between the workload size and the resulting index size, with an average ratio of 1:2. This is largely as a result of indexing all the 15 Dublin Core repeatable fields. In addition, all the record fields were stored in the index when conducting the experiment. In an ideal scenario, only relevant fields would have to be indexed and stored, significantly reducing the resulting index size.

The indexing operation throughput generally increases with increasing workload, reaching a peak value of 803 documents/second at workload $W9$—as shown in Figure 6-12, after which it plummets. This scenario is attributed to the fact that Apache Solr indexing is, in part, dependent on the size on the index—the index size linearly increases with workload size. Furthermore, the 2 GB RAM on the graduate becomes inadequate as the workload increases, thus degrading the performance.

Table 6-8. Baseline performance benchmarks for batch indexing

	Index [MB]	Time [s]	Throughput [doc/s]
W1	0.40	1.11	90
W2	0.56	1.47	136
W3	1.10	2.46	163
W4	1.90	4.14	193
W5	3.50	7.19	223
W6	6.40	13.25	242
W7	12.00	14.81	432
W8	24.00	17.89	715
W9	46.00	31.87	803
W10	91.00	204.21	251
W11	179.00	432.12	237
W12	348.00	1331.99	154
W13	962.00	2934.96	140
W14	1433.60	8134.99	101
W15	2662.40	18 261.88	90

Experiment: Incremental collection indexing benchmarks This experiment was conducted to assess the performance of the indexing process, relative to the size of the collection, when collections are updated with new content.

Methodology A batch of 1000 latest records[8] were harvested from the NDLTD portal and added to existing workload indices using Apache Solr XSLT UpdateRequestHandler (*Solr Wiki* 2012b). The number of documents added to the indices was varied between 1, 10, 100 and 1000. In addition, the changes were only committed to the indices after all records had been added to the index.

[8]OAI-PMH 'from' parameter was used to harvest records not previously harvested

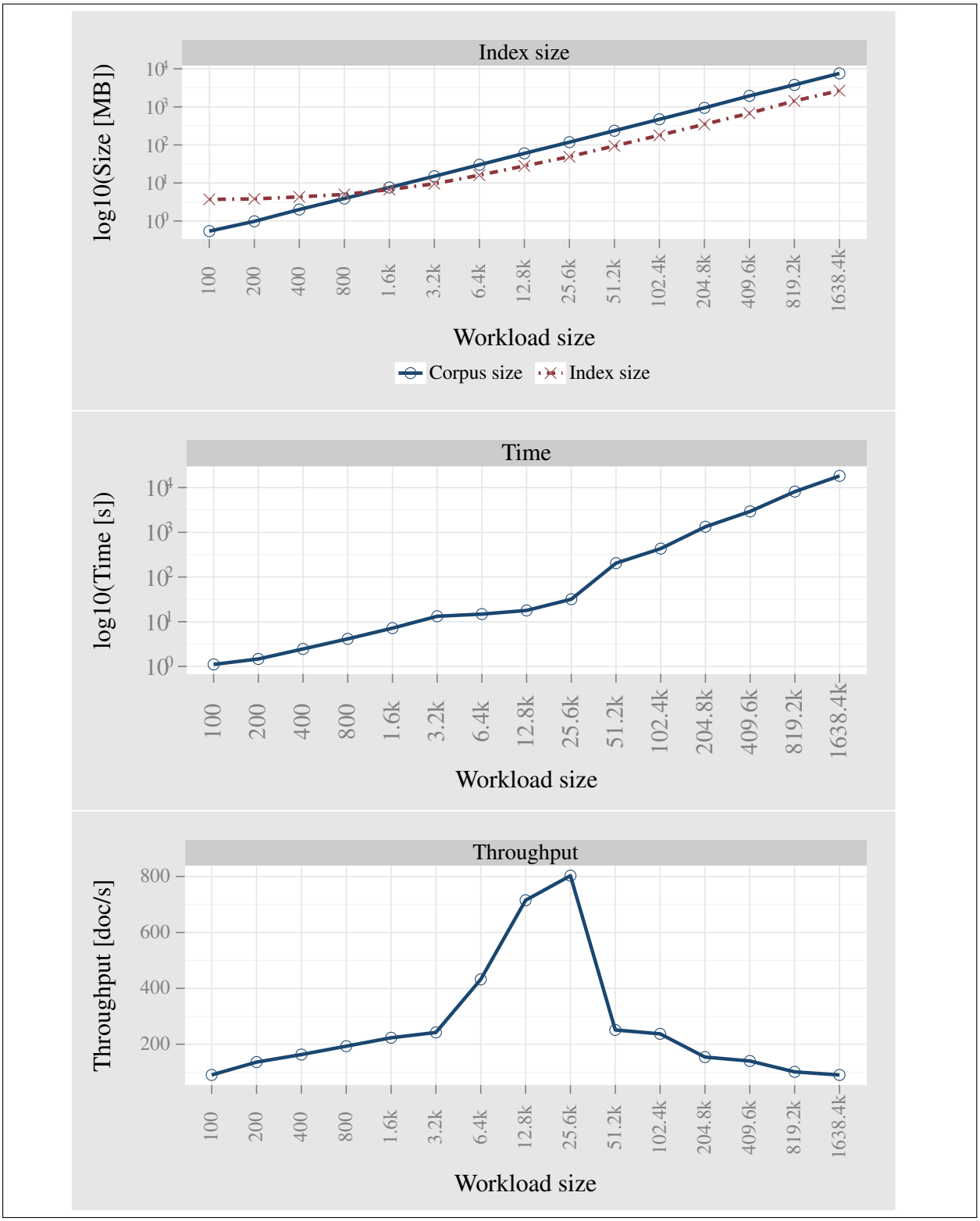


Figure 6-12. Indexing operations performance benchmarks results showing the size of the indices on disk, the time taken to generate the indices, and the indexing process throughput. Notice how the throughput plummets when the workload goes beyond 25 600 documents.

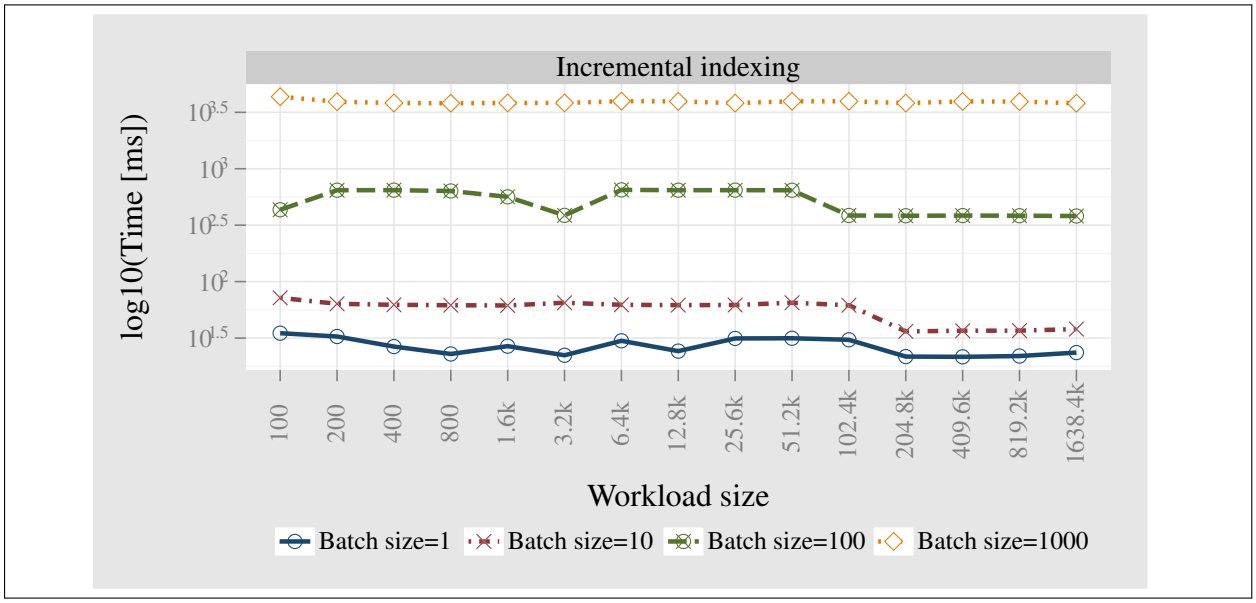


Figure 6-13. Incremental Index documents update operations performance benchmarks.

Results Table 6-9 and Figure 6-13 show the experiment results.

Table 6-9. Impact of batch size on indexing performance

	Batch size							
	1		**10**		**100**		**1000**	
	Indexing	**Parsing**	**Indexing**	**Parsing**	**Indexing**	**Parsing**	**Indexing**	**Parsing**
W1	88.78 %	11.22 %	45.96 %	54.04 %	45.04 %	54.96 %	39.44 %	60.56 %
W2	88.89 %	11.11 %	39.34 %	60.66 %	40.44 %	59.56 %	39.08 %	60.92 %
W3	90.32 %	9.68 %	40.13 %	59.87 %	42.40 %	57.60 %	40.60 %	59.40 %
W4	87.52 %	12.48 %	35.64 %	64.36 %	41.08 %	58.92 %	39.95 %	60.05 %
W5	85.85 %	14.15 %	47.19 %	52.81 %	42.56 %	57.44 %	39.61 %	60.39 %
W6	89.87 %	10.13 %	41.56 %	58.44 %	39.86 %	60.14 %	40.07 %	59.93 %
W7	90.35 %	9.65 %	41.74 %	58.26 %	42.77 %	57.23 %	43.20 %	56.80 %
W8	90.86 %	9.14 %	45.26 %	54.74 %	43.07 %	56.93 %	39.31 %	60.69 %
W9	92.49 %	7.51 %	41.92 %	58.08 %	42.09 %	57.91 %	40.25 %	59.75 %
W10	88.96 %	11.04 %	41.57 %	58.43 %	42.65 %	57.35 %	42.01 %	57.99 %
W11	88.53 %	11.47 %	37.27 %	62.73 %	42.29 %	57.71 %	42.92 %	57.08 %
W12	87.71 %	12.29 %	38.68 %	61.32 %	41.74 %	58.26 %	39.96 %	60.04 %
W13	88.15 %	11.85 %	40.87 %	59.13 %	39.52 %	60.48 %	42.76 %	57.24 %
W14	86.77 %	13.23 %	32.63 %	67.37 %	41.71 %	58.29 %	42.72 %	57.28 %
W15	89.37 %	10.63 %	36.89 %	63.11 %	38.05 %	61.95 %	40.15 %	59.85 %

Discussion The conversion process of records to Apache Solr ingest format takes up a considerable amount of time during parsing. In addition, it is significantly faster to schedule commits for large sets of newer records in contrast to issuing commits after addition of each record, since the cumulative commit times for individual items in a typical batch are avoided.

OAI-PMH data provider

The main objective of experiments associated with this aspect was to determine the performance of an integrated file-based collection OAI-PMH data provider in relation to the collection size.

The XMLFile Perl data provider module (Suleman, 2002) was used to conduct the experiments. The module was configured and deployed within a mod_perl[9] enabled Apache 2.2.22 Web server. The following factors were considered, relative to the workloads described in Section 6.2.3.

- The collection structure
- The size of the resumptionToken

Experiment: OAI-PMH data provider baseline benchmarks This experiment was conducted to derive baseline results for a basic OAI-PMH data provider environment set up.

Methodology The OAI-PMH data provider for each archive was configured with a resumptionToken of 1000 and the records in each workload arranged in a one-level hierarchical structure, as shown in Figure 6-8a

The tests performed involved submitting GetRecord, ListIdentifiers, ListRecords, and ListSets requests to each of the individual 15 workloads. Siege was used to to simulate a single user request with a total of 5 repeated runs for each request; the average response times for each case were then recorded.

Results The response times for the four OAI-PMH verbs are shown in Figure 6-14.

Discussion The ListRecords and ListIdentifiers verbs are the most expensive of the OAI-PMH verbs, each taking more than 2 seconds when the workload size goes beyond 400 and 6400 respectively. In contrast, the GetRecord and ListSets verbs only go beyond acceptable limits when the workload size exceeds 204 800 and 819 200 respectively.

Experiment: Impact of collection structure The results obtained from the baseline experiment conducted in Experiment 1 involved the use of a one-level collection structure illustrated in Figure 6-8a. This experiment was conducted to assess the impact that a multi-level structure would have on the overall performance of an OAI-PMH data provider whilst varying the workload.

[9] An Apache/2.x HTTP server embedded Perl interpreter

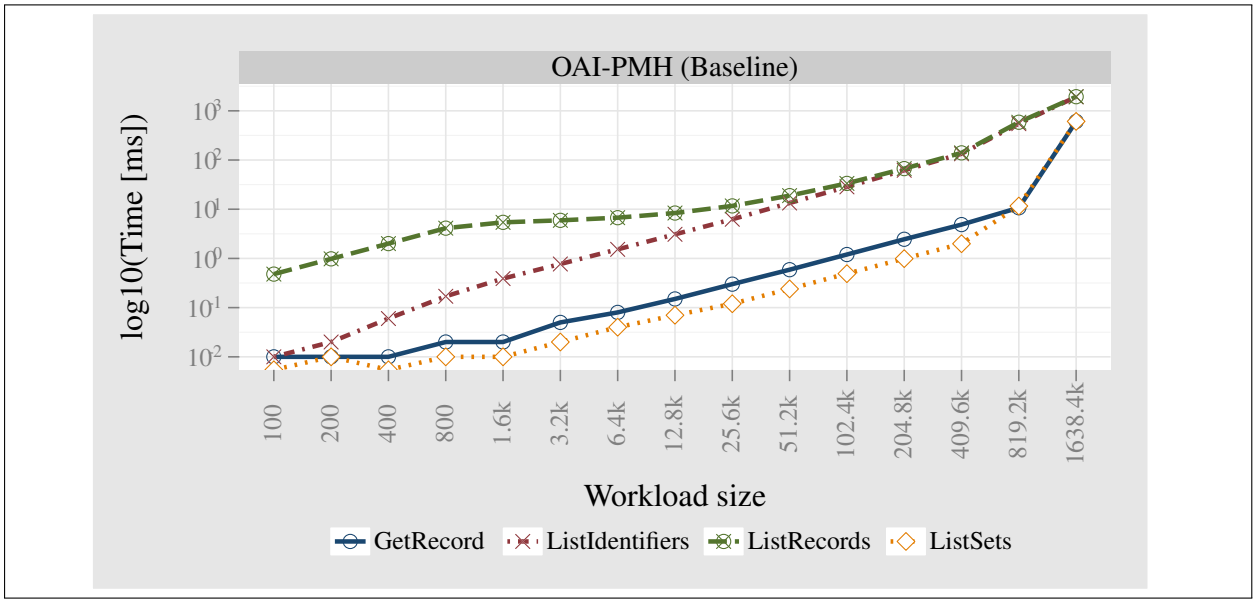


Figure 6-14. OAI-PMH data provider baseline performance benchmarks results for all four request verbs.

Methodology A three-level collection structure, shown in Figure 6-8c, was used. Siege was then used to to simulate a single user request with a total of 5 repeated runs for GetRecord, ListIdentifiers, ListRecords and ListSets verbs; the average response times for each case were then recorded.

Results Figure 6-15 show results of the impact on performance of collection structure on the OAI-PMH verbs.

Discussion The difference in response times, of ListIdentifiers and ListRecords verbs, for the different levels only becomes apparent with relatively larger workloads. This difference is as a result of the latency incurred during directory traversal, an operation that takes a relatively shorter time to complete. This is further evidenced by the results from the ListSets verb (see Figure 6-15), an operation that is significantly dependent on directory traversal.

Experiment: Impact of resumptionToken size The flow control for incomplete list responses in Experiment 1 was handled based on the recommendations from the guidelines for repository implementers (Lagoze et al., 2002a). This involved the use of a resumptionToken size of 1000 records. This experiment was conducted to determine the impact of varying the resumptionToken sizes as the workload increased.

Methodology The resumptionToken sizes were varied between 10, 100 and 1000, whilst conducting ListIdentifiers and ListRecords requests for the first and last list responses. Siege was used to simulate a single user request with a total of 5 repeated runs for each request; the average response times for each case were then recorded.

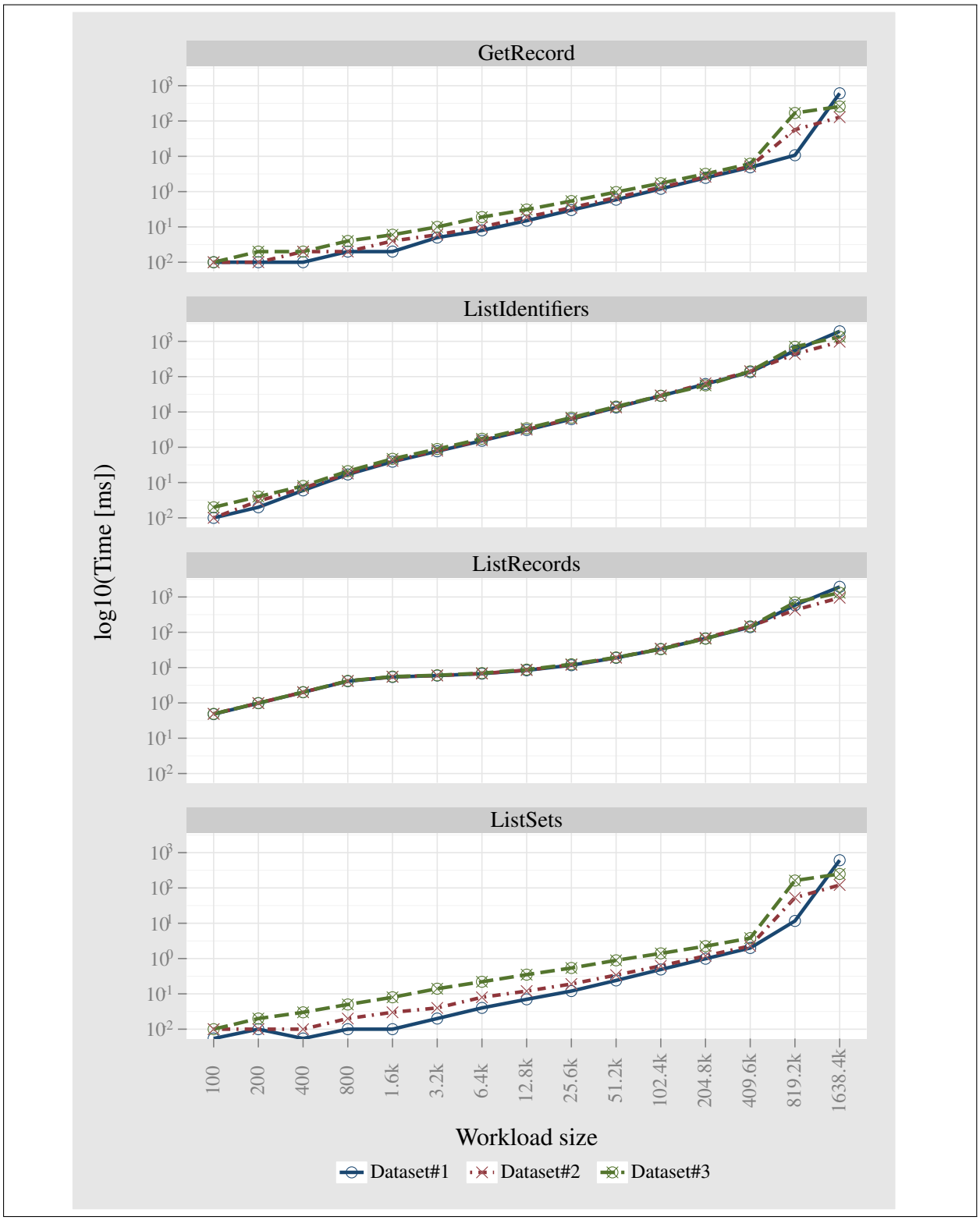


Figure 6-15. Impact of collection structure on OAI-PMH data provider performance. Note that with the exception of workloads with less than 1000 documents, ListIdentifiers and ListRecords are partial incomplete-list responses for the first N=1000 records.

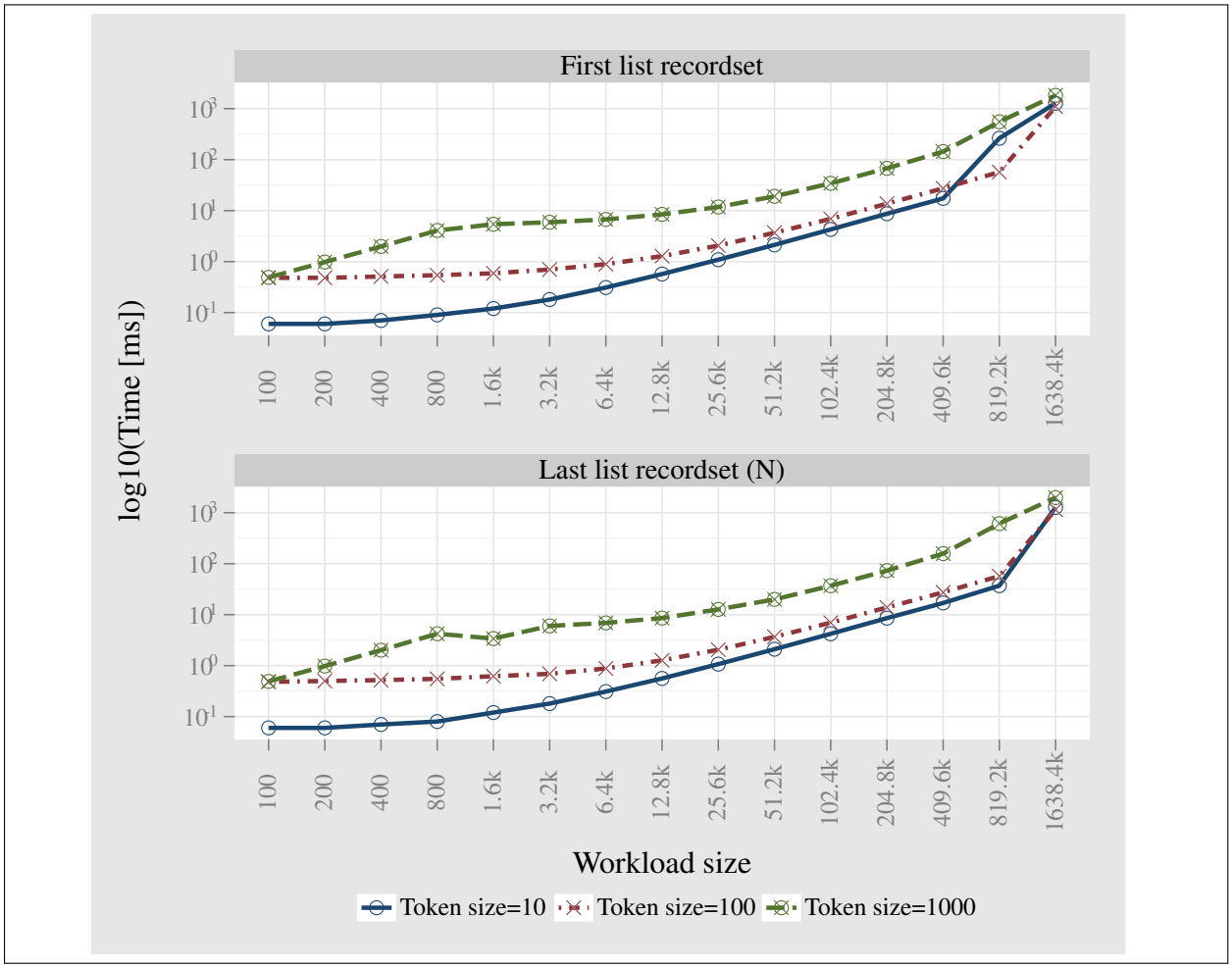


Figure 6-16. Impact of resumptionToken size on OAI-PMH data provider performance. The plots show the time taken to generate archives' first partial incomplete list set and the archives' last list set.

Results The results of the experiment are shown in Figure 6-16 in the form of response times for ListRecord verb when resumptionToken size is 0 (First list recordset) and when resumptionToken size is N (Last list recordset); with N representing the last list response.

Discussion The results indicate that harvesting recordsets using a smaller resumptionToken size is faster than otherwise. In addition, there is not a noticeable change when the resumptionToken cursor is varied.

Experiment: Impact of resumptionToken size and structure This experiment was conducted to assess the combined effect of a structured collection and varying resumptionToken sizes.

Methodology The collection structures shown in Figure 6-8 was used and resumptionToken sizes varied as in the experiment described in Section 6.2.4. Siege was then used to compute response times for generating incomplete list responses for the ListRecords OAI-PMH verb.

Results Figure 6-17 shows results of the combined effect of hierarchical structure and resumptionToken size.

Discussion The results indicate that it is significantly faster to harvest records from workloads with fewer hierarchical levels and at the same time smaller resumptionToken sizes. The reason for this is two-fold: first, the traversal time for workloads with fewer levels is reduced; and secondly, the time taken to sort records with smaller resumptionToken sizes is faster.

Feed generator

The purpose of this experiment was to determine how the relative size of file-based collections could potentially impact the performance of an integrated RSS feed generator module.

Experiment: Impact of collection structure This experiment was conducted to investigate the performance of a file-based RSS module for non-indexed collections.

Methodology The approach used to determine top N latest records took advantage of the operating system creation and modification timestamps. This approach was used to avoid the overhead that results from parsing individual records. Each of the 15 workloads were traversed to determine the response times when generating top N latest records.

This technique was repeated for one-level, two-level and three-level hierarchical structures and using results from one-level structures as the baseline, the change in response times was noted to determine the effect of altering the collection structures.

Results Figure 6-18 shows the times taken to generate the top N most recently added records to each of the 15 workloads. Table 6-10 and Figure 6-19 show the change (Δ Dataset#2 and Δ Dataset#3) in response times—relative to one-level structured workloads results shown in Table B-14—for each of the workloads when rearranged into two-level and three-level structures.

Table 6-10. Impact of structure on feed generation

	Δ Dataset#2			Δ Dataset#3		
	5	**10**	**20**	**5**	**10**	**20**
W1	63.85 %	61.62 %	58.54 %	152.75 %	151.05 %	145.19 %
W2	58.16 %	54.64 %	54.32 %	150.73 %	146.84 %	141.73 %
W3	51.97 %	47.85 %	49.57 %	141.99 %	140.05 %	136.78 %
W4	41.63 %	37.10 %	38.98 %	125.78 %	123.37 %	121.50 %
W5	33.92 %	31.56 %	33.01 %	114.05 %	110.42 %	112.09 %
W6	23.49 %	23.24 %	21.40 %	93.80 %	94.39 %	106.34 %
W7	17.30 %	14.93 %	17.00 %	81.40 %	76.52 %	79.33 %

(Continued on next page)

Table 6-10. (continued)

	Δ Dataset#2			Δ Dataset#3		
	5	**10**	**20**	**5**	**10**	**20**
W8	13.54 %	12.53 %	11.65 %	60.91 %	61.63 %	67.42 %
W9	8.19 %	9.78 %	8.34 %	54.52 %	52.73 %	50.33 %
W10	6.31 %	6.47 %	4.88 %	34.99 %	44.35 %	40.90 %
W11	4.12 %	5.91 %	6.13 %	26.02 %	26.94 %	34.42 %
W12	3.69 %	4.28 %	4.14 %	55.52 %	64.81 %	57.05 %
W13	9.00 %	8.98 %	9.24 %	101.65 %	94.27 %	100.57 %
W14	(50.45) %	(55.47) %	(57.92) %	38.85 %	25.31 %	18.00 %
W15	(90.14) %	(90.06) %	(90.36) %	(80.48) %	(80.50) %	(80.84) %

Discussion The results presented in Figure 6-18 indicate that there is not a noticeable change in the overall performance when the collection structure is changed. This is primarily due to the fact that the only significant factor involved during feed generation is the comparison of metadata file timestamps—an operation which is very efficient. Another significant factor involved in the feed generation process is directory traversal time, which remains almost constant for varying feed sizes, since the structure remains unchanged. However, increasing the feed sizes to larger sizes would result in some noticeable variation, since the time for comparing file timestamps would be increased significantly.

Table 6-10 and Figure 6-19 show a noticeable change in the response times for two-level and three-level structured workload collections, relative to one-level structured workloads. This change is as a result of the increase in the traversal times as the hierarchies are increased.

6.2.5 Comparisons

DL scalability (Misra, Seamans, and Thoma, 2008) and stress-testing (Bainbridge et al., 2009) experiments conducted in the past have mostly been focused on specific aspects of DLSes. However, some comparative studies (*Fedora Performance and Scalability Wiki* 2012) have been conducted, specifically aimed at gathering data used to make improvements to tools and services.

The comparative experiments conducted are similar to the work presented by Misra et al. (Misra, Seamans, and Thoma, 2008). However, as opposed to the ingest-focused benchmarks they conducted, the results presented in this section involved varying aspects of DLs. In addition, they were specifically conducted to compare two different approaches—the simple repository design and DSpace 3.1 (*DSpace Wiki* 2013).

Methodology

A total of 15 DSpace 3.1 instances were set up corresponding to the 15 experiment workloads. The community and collection hierarchies corresponding to workload name and $setSpecs$ respectively,

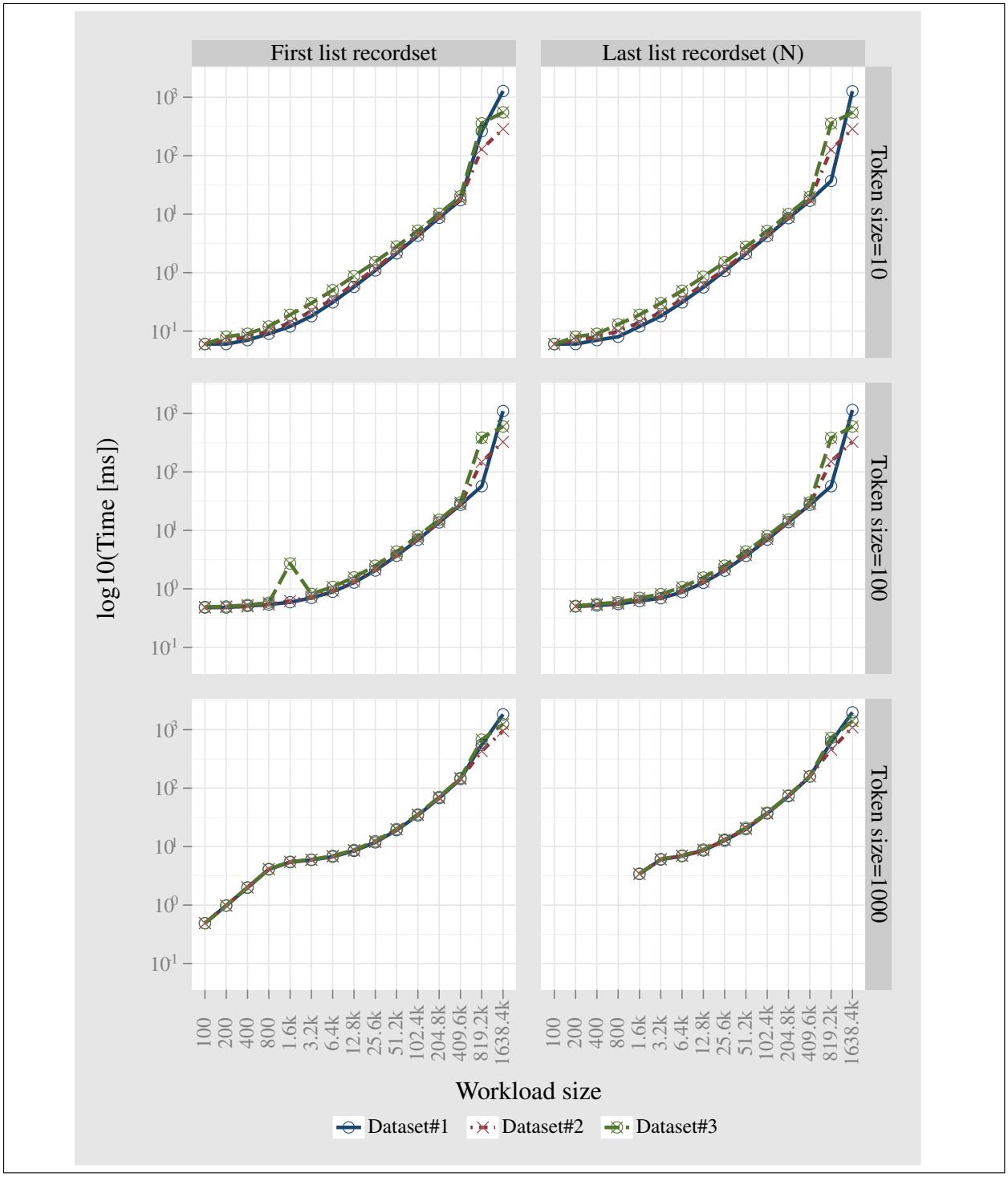


Figure 6-17. Impact of varying resumptionToken sizes and collection structure on the OAI-PMH Data Provider performance.

Figure 6-18. Impact of feed size on feed generation performance

in each of the 15 workloads, were then pre-created; this process was necessitated by the fact that item ingestion within DSpace can only be performed on existing collections.

The following evaluation aspects were then performed on the DSpace instances and subsequently compared with performance results from workloads ingested into the 15 file-based repositories.

- Item ingestion performance
- Search query performance
- OAI-PMH data-provider performance

Results

Figure 6-20 is a comparison of the ingest response times for a potential non-indexed file-based repository and DSpace; Figure 6-22 show OAI-PMH comparisons; and Figure 6-21 show the comparison of search query performance between the two approaches.

Discussion

Figure 6-20 shows that the average time taken to ingest a single item using the proposed approach is significantly more efficient than DSpace. Furthermore, the ingest time generally remains constant as the workload is increased. The reason for this is that parsing and repository disk write are the only ingest phases required to ingest an item into the repository, with parsing and disk writes accounting for 90 % and 10 % of the total ingest time respectively. In contrast, the DSpace ingest phase comprises of an item-level database write phase (org.dspace.content.Item),

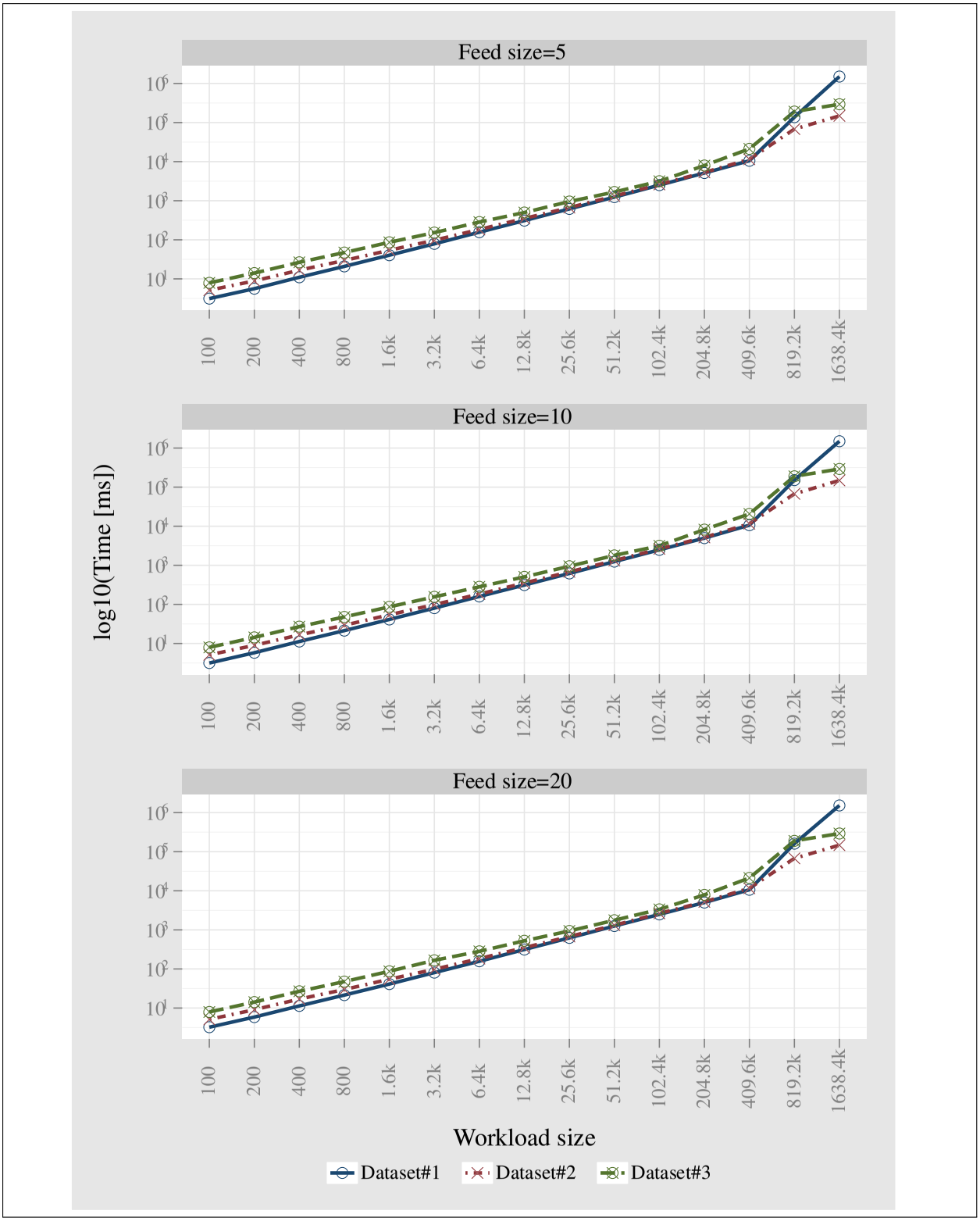


Figure 6-19. Impact of structure on feed generation performance

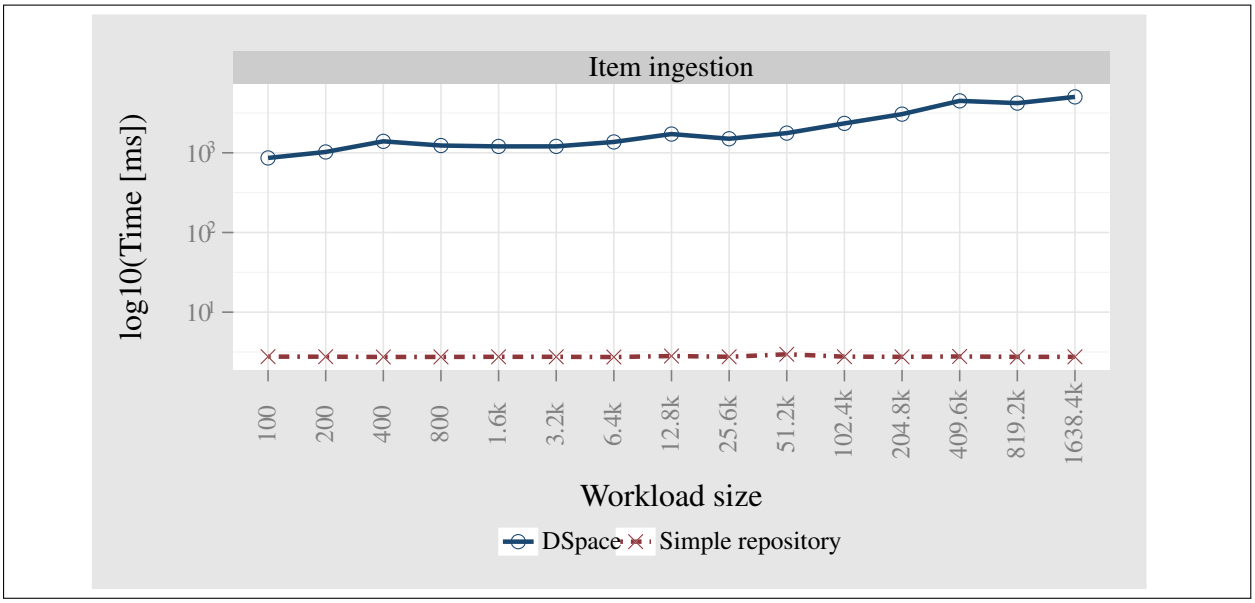


Figure 6-20. A plot showing a comparison of ingestion performance between the simple repository and DSpace.

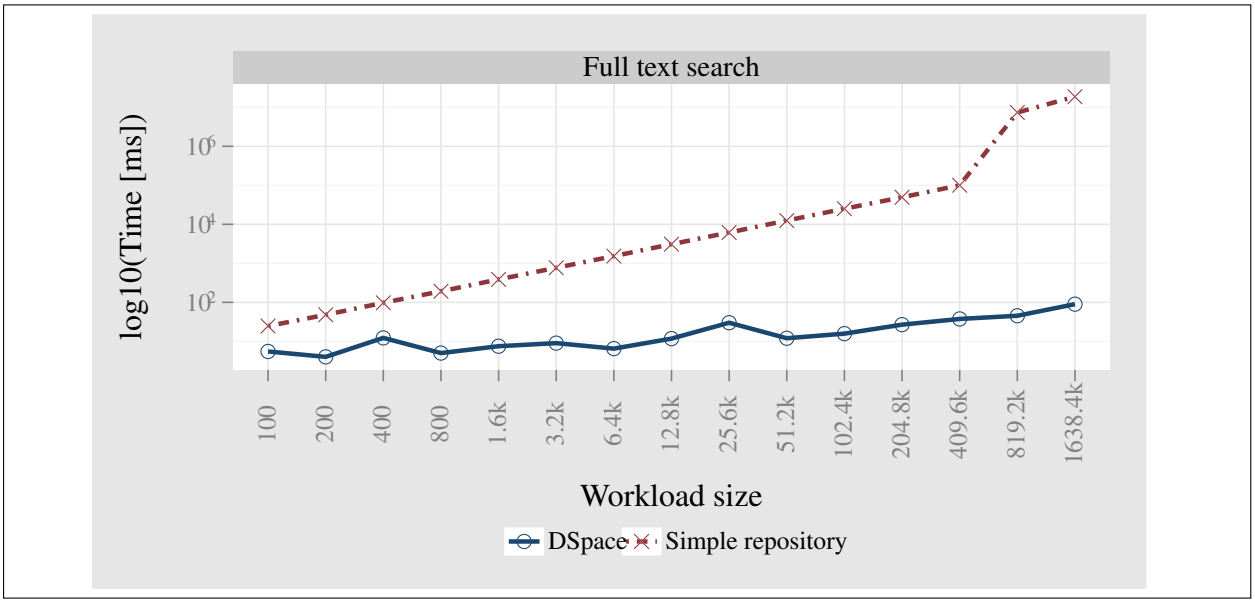


Figure 6-21. A plot showing a comparison of full-text search performance between the simple repository and DSpace.

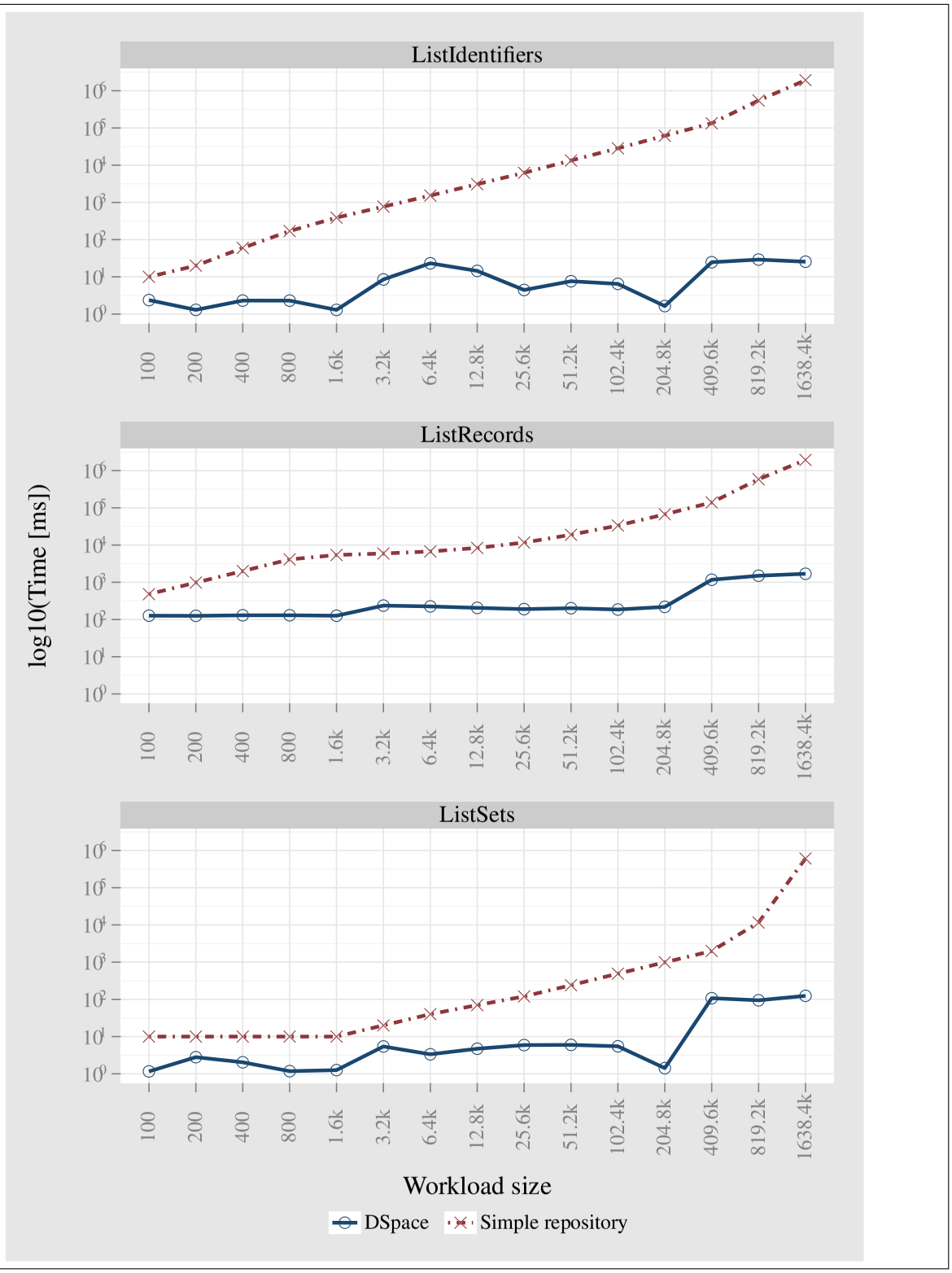


Figure 6-22. A plot showing a comparison of OAI-PMH performance between the simple repository and DSpace.

a collection-level database write phase (org.dspace.content.Collection) and an indexing phase (org.dspace.search.DSIndexer).

Search operations and OAI-PMH data provider operations, shown in Figure 6-21 and Figure 6-22, are orders of magnitude faster on DSpace in comparison to a file-based store. The response times on DSpace for these operations are significantly a result of a third-party search service (Apache Solr) integrated with the application to facilitate fast search. The uneven plots—top and bottom plots corresponding to DSpace—in Figure 6-22 are as a result of the difference in the structure of the metadata records from the different collections—the DSpace instances used in the experiments were configured using an OAI 2.0 data provider that uses a Solr data source by default.

These findings suggest that comparable speeds could be easily attained if the file-based repository was integrated with a search service. Incidentally, integration of a file-based repository with a search service was shown to be possible in 6.2.4.

6.2.6 Summary

The results of the performance experiments helped confirm the following:

- The proposed simple repository design yields acceptable performance for relatively medium-sized unindexed collections.
- The comparative experiments with DSpace indicate that—comparable performance can be achieved if the simple repository were to be integrated with a third-party search service.
- The majority of operations would be dependent on parsing for unindexed collections.

6.3 Summary

The results from the developer survey (see Section 6.1) have shown that the resulting simple file-based repository design is easy to work with and could potentially simplify repository management tasks. Furthermore, the results also indicate that a simple file-based repository design would have little impact on the extensibility of an application built on top of such a repository design.

The implementation case study collections outlined in Section 5 serve as proof that the proposed approach is effective; the Bleek and Lloyd case study (see Section 5.1) in particular serves as proof that system functionalities and features of existing services based on conventional storage solutions can be replicated using a simple file-based digital object store with little adverse effect.

The scalability performance experiments yielded results that strongly indicate that the performance would be within generally acceptable limits for medium-sized collections, as evidenced in the Kiviat plot shown in Figure 6-23. Figure 6-23 also indicates that ingestion performance is significantly better than the other services. In addition, the performance degradation for all other services occurs for collections with larger than 12 800 objects. It was further shown that performance degradation of operations such as information discovery and OAI-PMH associated services are largely as a result of parsing, a problem that can easily be remedied through the use of an index.

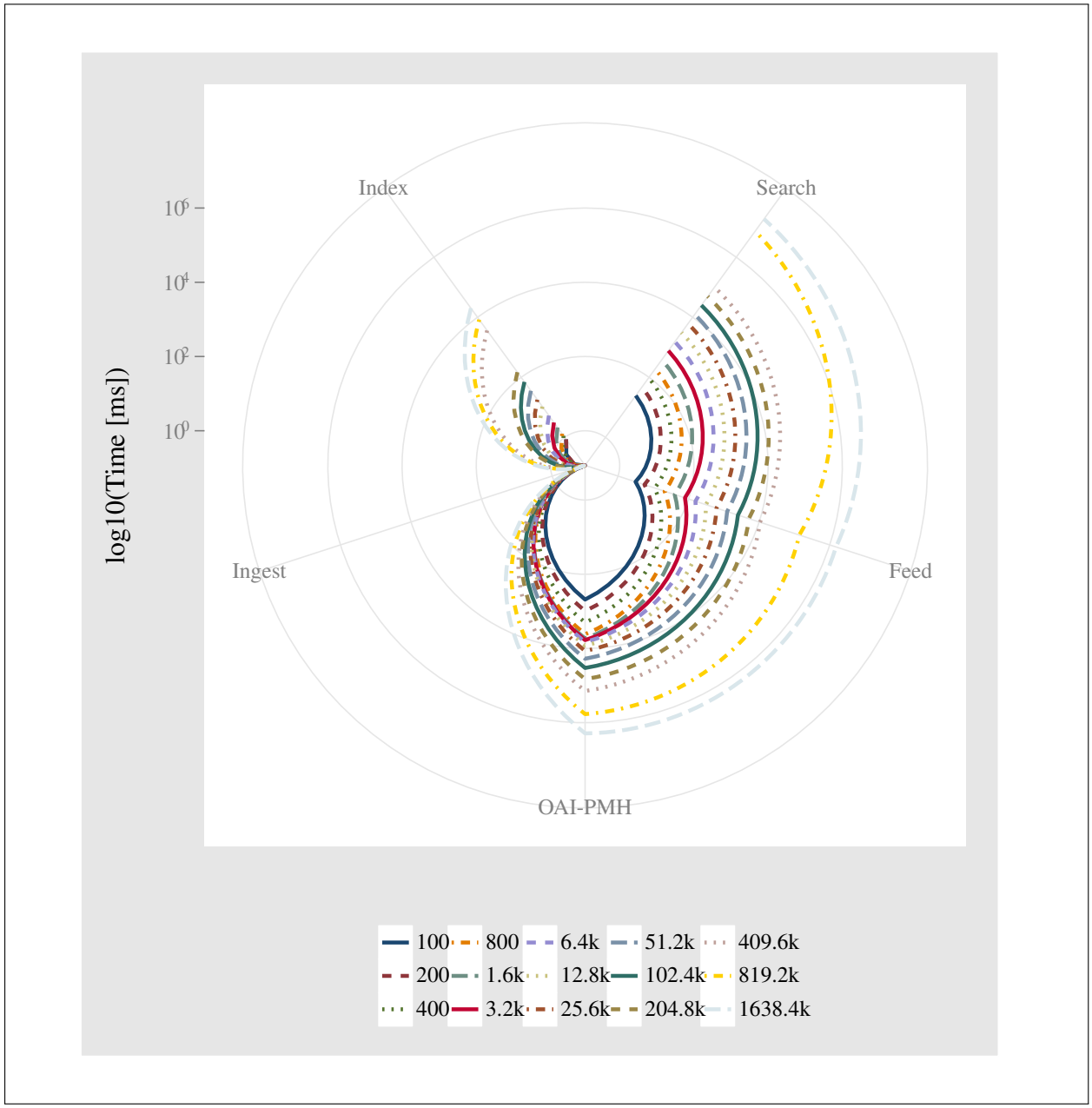


Figure 6-23. A Kiviat plot showing performance degradation (increase in response times) of evaluation aspects—batch indexing, full-text search, OAI-PMH data provider, RSS feed generator and single item ingestion—relative to increasing workload sizes, with each polar line representing the 15 experiment workloads.

Finally, it was shown that the superior performance results from the comparative experiments done with DSpace are attributed to the external index service—Apache Solr and Lucene—integrated with DSpace to facilitate fast search. However, it was shown in the indexing experiments (see Section 6.2.4) that integration of such an external search service could easily be performed using the proposed approach.

Chapter 7

Conclusions

This research was motivated by the observation that most contemporary DL tools are complex and thus difficult to manage. The design of simpler and more manageable tools for storage and management of digital content was subsequently identified as a potential solution. A literature synthesis of the two-decades long study of DLs suggests that there is now a firm understanding of the basic underpinning concepts associated with DLSes. This is evident from the varying existing designs of tools and services specifically tailored to store, manage and preserve digital content. In Chapter 2, some prominent DL frameworks and software tools were presented to illustrate the differences in the design approach. Furthermore, the relevant background information was also presented.

An exploratory study, discussed in Chapter 3, was conducted using Grounded Theory as the overarching research method to help derive a set of guiding design principles that would aid the overall design of simple DLSes. A practical application of the guiding principles, discussed in Chapter 4, was assessed through the design of a simple repository sub-layer for a typical DLS and the effectiveness of the design subsequently evaluated through the implementation of two real-world case studies that are discussed in Chapter 5. In addition to assessing the effectiveness of this research through the case studies implementations, a developer survey (see Section 6.1) was conducted to assess the simplicity and usefulness of the approach. Finally, a series of performance benchmarks, discussed in Section 6.2, were conducted to assess the implications of simplifying DLS design relative to the collection size.

7.1 Research questions

The research questions that were formulated at the onset of this research, as described in Section 1.3 were addressed through the exploratory study discussed in Chapter 3; the prototype repository design described in Chapter 4; the case study implementation presented in Chapter 5; and through experiments outlined in Chapter 6. In summary, the research questions were resolved as follows:

Is it feasible to implement a DLS based on simple architectures?

The prototype repository design in Chapter 4, together with the real-world case study implementations discussed in Chapter 5 prove the feasibility of simple designs. This assertion is further supported by the various Web services that were developed during the developer study described in Section 6.1.

i **How should simplicity for DLS storage and service architectures be defined?**

A major outcome, and perhaps a significant contribution of this research revolves around a principled approach to simple DLS design. This approached offers the advantage of ensuring that domain and application-specific needs are met. Furthermore, such a principled design approach could have potential practical application to other applications, other than DLSes, with distinct domain-specific needs. This outcome was implicitly derived as a direct manifestation of results from the research questions discussed in Chapter 1.

ii **What are the potential implications of simplifying DLS—adverse or otherwise?**

The results from the developer survey suggest that the proposed approach does not adversely impact the overall extensibility of the prototype repository design. This inference is supported by the varying implementation languages and techniques utilised by the survey participants. In addition, only four out of the 12 groups used additional back-end tools to develop a layered service on top of the simple repository sub-layer used in the survey.

iii **What are some of the comparative advantages and disadvantages of simpler architectures to complex ones?**

The results from the performance-based experiments indicate that the performance of information discovery operations relative to the size of the collection is adversely impacted; the results show that a collection size exceeding 12 800 items results in an response times exceeding 10 seconds for certain DLS operations. However, owing to the fact that the affected operations are information discovery related, this shortcoming can be resolved by integrating the DLS with an indexing service. Interestingly, ingest-related experiments resulted in superior response times.

7.2 Future work

The objectives of this research were successfully achieved. However, there are still a number of potential research directions that could be further explored. The following are some potential future research areas that could be explored to complement the work conducted in this research.

7.2.1 Software packaging

A key issue that has been linked to user adoption and overall usability of DL software is the installation and configuration process associated to such systems (Körber and Suleman, 2008). There have been a number of attempts to implement out-of-the-box systems (Maly et al., 2004; *Installing EPrints 3 via apt (Debian/Ubuntu)* 2011). However, these have mostly been specific to particular operating system platforms. A potential research area could thus involve investigating how to simplify the packing of DL tools and services.

7.2.2 Version control

The integration of digital object version control could significantly complement the preservation of resources stored in DLs. This is an area that is already currently being explored (*Item Level Versioning* 2012). However, there is still a need to further explore how this important aspect of DL preservation can be simplified.

7.2.3 Reference implementation

The applicability of the design principles was presented in form of a simple prototype repository design. However, DLSes are multi-faceted applications and it would be interesting to design and implement a reference implementation composed of components—user interface and service layer components—designed using this prescribed design approach. This would further set the stage to conduct user studies aimed at determining whether simplifying the overall design of DLSes would have an impact on the way users interact with such systems. In addition, this would make it possible for desirable aspects of DLs, for instance interoperability, to be evaluated as part of a complete system. Furthermore, a detailed evaluation of the integration of prominent DLS-specific standards and protocols with such a reference implementation would prove invaluable.

Appendix A

Developer survey

This appendix provides extra information related to the developer user study outlined in Section 6.1. Ethical clearance related information is outlined in A.1 and questionnaire design related information in A.2.

A.1 Ethical clearance

The University of Cape Town has assigned ethics clearance authority to Faculty-level Research Ethics Committees. In addition, permission to access staff and student target populations, as research participants, is assigned to the Executive Director: Human Resource and the Executive Director: Student Affairs respectively. In a nutshell, the ethics clearance standard operating procedure ensures that;

- Permission to access staff or student populations must be obtained from ED: HR for staff and ED: Student Affairs for students.
- This process is separate from the ethics clearance process.
- Ethics clearance must be sought from the Faculty-level Research Ethics Committee in the Faculty closest to the area of research proposed.
- The proposed research may proceed only when both permission to access and ethics clearance have been obtained.

This appendix section provides screenshots of ethical clearance obtained prior to undertaking user studies. Figure A-1 is a screenshot of permission received from the science faculty to undertake the user study and Figure A-2 is a screenshot of approval obtained from Student Affairs to use university students as subjects for the user study.

Department of Environmental and Geographical Science
University of Cape Town
RONDEBOSCH 7701
South Africa

e-mail: Michael.meadows@uct.ac.za
phone : + 27 21 650 2873
fax : +27 21 650 3791

21st May 2012

Mr Lighton Phiri
Department of Computer Science
University of Cape Town
lphiri@cs.uct.ac.za

Dear Mr Phiri

A Lightweight Digital Library Framework

I am pleased to inform you that, having scrutinized the details of your above-named application for research ethics clearance, the Faculty of Science Research Ethics Committee has approved it in terms of its attention to ethical principles.

Your approval code is: SFREC 011_2012

I wish you success in the work involved.

Yours sincerely

M Meadows

Michael E Meadows
Professor and Head of Department
Chair: Science Faculty Ethics in Research Committee

Figure A-1. Screenshot of faculty research ethical clearance

	RESEARCH ACCESS TO STUDENTS	DSA 100

NOTES

1. This form must be FULLY completed by applicants that want to access UCT students for the purpose of research.
2. Return the completed application form together with your research proposal to: Moonira.Khan@uct.ac.za; or deliver to: Attention: Executive Director, Department of Student Affairs, North Lane, Steve Biko Students' Union, Room 7.22, Upper Campus, UCT.
3. The turnaround time for a reply is approximately 10 working days.
4. NB: It is the responsibility of the researcher/s to apply for and to obtain ethical clearance and access to staff and/or students, respectively to the (a) Faculty's 'Ethics in Research Committee' (EiRC) for ethics approval, and (b) Executive Director, HR for approval to access staff for research purposes and the (c) Executive Director, Student Affairs for approval to access students for research purposes.
5. For noting, a requirement of UCT (according to Senate policy) is that items (1) and (4) apply even if prior clearance has been obtained by the researcher/s from any other institution.

SECTION A: RESEARCH APPLICANT/S DETAILS

Position	Staff / Student No	Title and Name	Contact Details (Email / Cell / land line)
A.1 Student Number	PHRLIG001	Mr Lighton Phiri	lphiri@cs.uct.ac.za
A.2 Academic / PASS Staff No.			
A.3 Visiting Researcher ID No.			
A.4 University at which a student or employee	UCT	Address if *not* UCT:	
A.5 Faculty/ Department/School	Science		
A.6 APPLICANTS DETAILS If different from above	Title and Name	Tel.	Email

SECTION B: RESEARCHER/S SUPERVISOR/S DETAILS

Position	Title and Name	Tel.	Email
B.1 Supervisor	A/Prof Hussein Suleman	+27 21 650 5106	Hussein@cs.uct.ac.za
B.2 Co-Supervisor/s (a)			

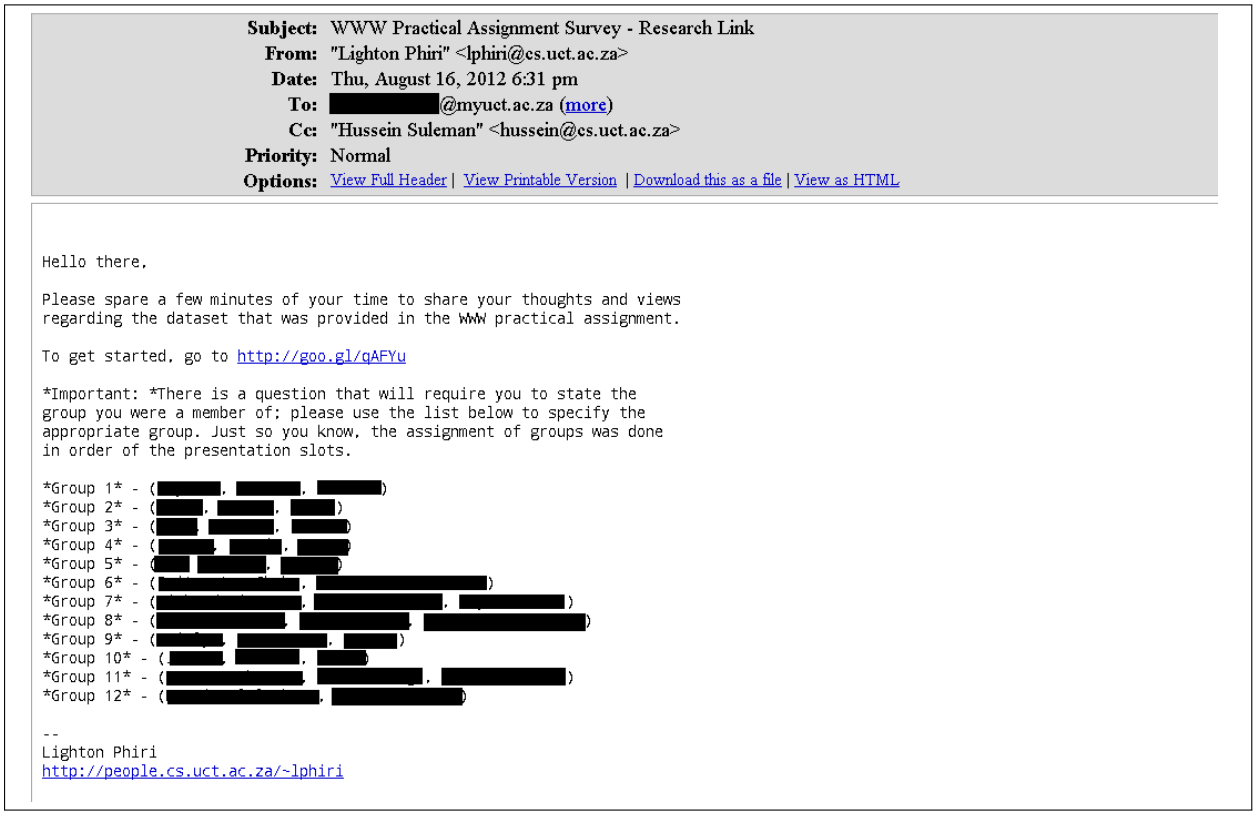
SECTION C: APPLICANT'S RESEARCH STUDY FIELD AND APPROVAL STATUS

C.1 Degree (if a student)	Master of Sciences
C.2 Research Project Title	A Lightweight Digital Library Framework
C.3 Research Proposal	Attached: Yes ■ No ☐
C.4 Target population	UCT Undergraduate, Honours & Postgraduate Students
C.5 Lead Researcher details	If different from applicant:
C6. Will use research assistant/s	Yes ☐ No ■
C.7 Research Methodology and Informed consent:	User Study #1: Participants will be recruited via e-mail and social networking site groups and ask to confirm their willingness to participate in the study. Those that will confirm will be required to fill out an online consent form. User Study #2: A group of honours students will be recruited to develop third-party tools and services after filling out a consent form.
C.8 Ethics clearance status from UCT's Ethics in Research Committee (EiRC)	Approved by the EiRC: Yes ■ No ☐ Awaiting response : ☐ If yes, attach copy and state the date and ref. no of EiRC approval: 21/05/2012 SFREC 011_2012 Date of application if awaiting response:

SECTION D: APPLICANT/S APPROVAL STATUS FOR ACCESS TO STUDENTS FOR RESEARCH PURPOSE
(To be completed by the ED, DSA or Nominee)

APPROVAL STATUS	Approved	Applicant/s Ref. No.:	Comments:	
	Yes ■	*Ref No. MR LIGHTON PHIRI / PHRLIG001*		
APPROVED BY:	Designation	Name	Signature	Date
	Executive Director Department of Student Affairs	*MBM KHAN*		*29/05/2012*

Figure A-2. Screenshot of student access ethical clearance



Subject: WWW Practical Assignment Survey - Research Link
From: "Lighton Phiri" <lphiri@cs.uct.ac.za>
Date: Thu, August 16, 2012 6:31 pm
To: [redacted]@myuct.ac.za (more)
Cc: "Hussein Suleman" <hussein@cs.uct.ac.za>
Priority: Normal
Options: View Full Header | View Printable Version | Download this as a file | View as HTML

Hello there,

Please spare a few minutes of your time to share your thoughts and views
regarding the dataset that was provided in the WWW practical assignment.

To get started, go to http://goo.gl/qAFYu

*Important: *There is a question that will require you to state the
group you were a member of; please use the list below to specify the
appropriate group. Just so you know, the assignment of groups was done
in order of the presentation slots.

Group 1 - ([redacted])
Group 2 - ([redacted])
Group 3 - ([redacted])
Group 4 - ([redacted])
Group 5 - ([redacted])
Group 6 - ([redacted])
Group 7 - ([redacted])
Group 8 - ([redacted])
Group 9 - ([redacted])
Group 10 - ([redacted])
Group 11 - ([redacted])
Group 12 - ([redacted])

--
Lighton Phiri
http://people.cs.uct.ac.za/~lphiri

Figure A-3. Screenshot showing the WWW survey participation email invitation

A.2 Survey design

This appendix section provides auxiliary information related to the developer survey described in Section 6.1. Figure A-3 is a screenshot of the email sent to the target population inviting them to participate in the survey, Figure A-4 is a screenshot of the WWW practical programming assignment assigned to the target population, and Figures A-5, A-5, A-5, A-5 and A-5 are screenshots of the post-assignment online questionnaire used by survey participants.

WWW: Data-driven Web-based Services

Introduction

Develop a new and useful Web-based service for end-users based on the data of the Bleek and Lloyd Collection of Bushman notebooks and drawings.

Web applications are generally based and implemented using tiered architectures, using a database for storing content and application-specific workflow data at the lowest layer. However, the use of a database is not always necessary and some preservation-directed projects opt to use a carefully constructed file store instead. The Bleek and Lloyd Collection at http://lloydbleekcollection.cs.uct.ac.za/ is based on such a file-store approach.

Your task is to use the Bleek and Lloyd data to create a new service that may reside on the same server as the data. Ideally, your service should be composed of a middleware layer with a machine interface that is then consumed by the front-end user interface. For your assignment, you should make a copy of the data and host this on your own machine.

Examples of services include: innovative metadata- or content-based search services, visualisation services, data/content management system, annotations, mining and subscription services, serendipitous discovery tools (think public areas), automatic linking services, integration with external sites,

Data Format

The data store has the following format:

```
/archive/books/dorotheableeknotebooks.metadata

/archive/books/dorotheableeknotebooks/BC_151_A3_01.metadata

/archive/books/dorotheableeknotebooks/BC_151_A3_01/A3_1_000FRC0V.JPG

/archive/books/dorotheableeknotebooks/BC_151_A3_01/A3_1_000FRC0V.JPG.metadata

/archive/books/dorotheableeknotebooks/BC_151_A3_01/A3_1_000INFRPG.JPG

/archive/books/dorotheableeknotebooks/BC_151_A3_01/A3_1_000INFRPG.JPG.metadata

...

/archive/stories
```

In general:

- All metadata files are in XML.
- Every file or directory may have as associated metadata file with a .metadata extension.

In the Bleek and Lloyd collection:

- The books subcollection is the primary data source.
- The stories subcollection is a view of the data and contains links to the images in the books subcollection.

The data may be downloaded from: http://goo.gl/sq6lV

If you have any questions about the data, please contact Lighton Phiri.

Requirements

Your application must be available through a Web interface. Use any Web server and application technology.

You are required to set up a Web server on a machine in the Honours lab used by one of the group members to host your application. Your application must include a clean copy of the original data without modification.

The data may be downloaded from: http://goo.gl/sq6lV

You may work in groups of up to 3 students. Send email to hussein by 3 August with the names of your group members and what service you intend to develop.

Your submission will be marked at a public demonstration (9am, 15 august) and a private demonstration thereafter. The criteria for marking are as follows:

- Correct operation of service 30%
- Complexity 20%
- Relevant usefulness 20%
- Originality/Creativity 20%
- Public demonstration 10%

Research Element

The structured data format is part of a research project in the Digital Libraries Laboratory to simplify the architecture of digital archives for long-term preservation of data as well as flexibility in developing new or replacement services. At the end of the project, you will be approached by the MSc student (Lighton Phiri) who is working on this project to obtain your feedback on the use of the data. This is optional but your input will be very much appreciated.

Figure A-4. Screenshot showing the WWW practical programming assignment question

Data Driven Web-based Services

Dear Respondent,

Thank you very much for taking the time to be part of this study.

This survey forms an integral part of a study aimed at investigating the impact of simplifying the design of digital library services .

There are 20 questions in this survey.

A Note On Privacy
This survey is anonymous.
The record kept of your survey responses does not contain any identifying information about you unless a specific question in the survey has asked for this. If you have responded to a survey that used an identifying token to allow you to access the survey, you can rest assured that the identifying token is not kept with your responses. It is managed in a separate database, and will only be updated to indicate that you have (or haven't) completed this survey. There is no way of matching identification tokens with survey responses in this survey.

Participant Consent

This study has received approval from the Ethics in Research Committee of the Science Faculty at the University of Cape Town, and the survey will take approximately 15 minutes to complete.
If you do wish to participate in this survey, please be advised that it will be anonymous and confidential.

* **I agree with the terms and hereby consent to participate in the study.**

○ Yes
○ No

Section 1 - Background Information

* **Which programming assignment group were you a member of?**

Choose one of the following answers

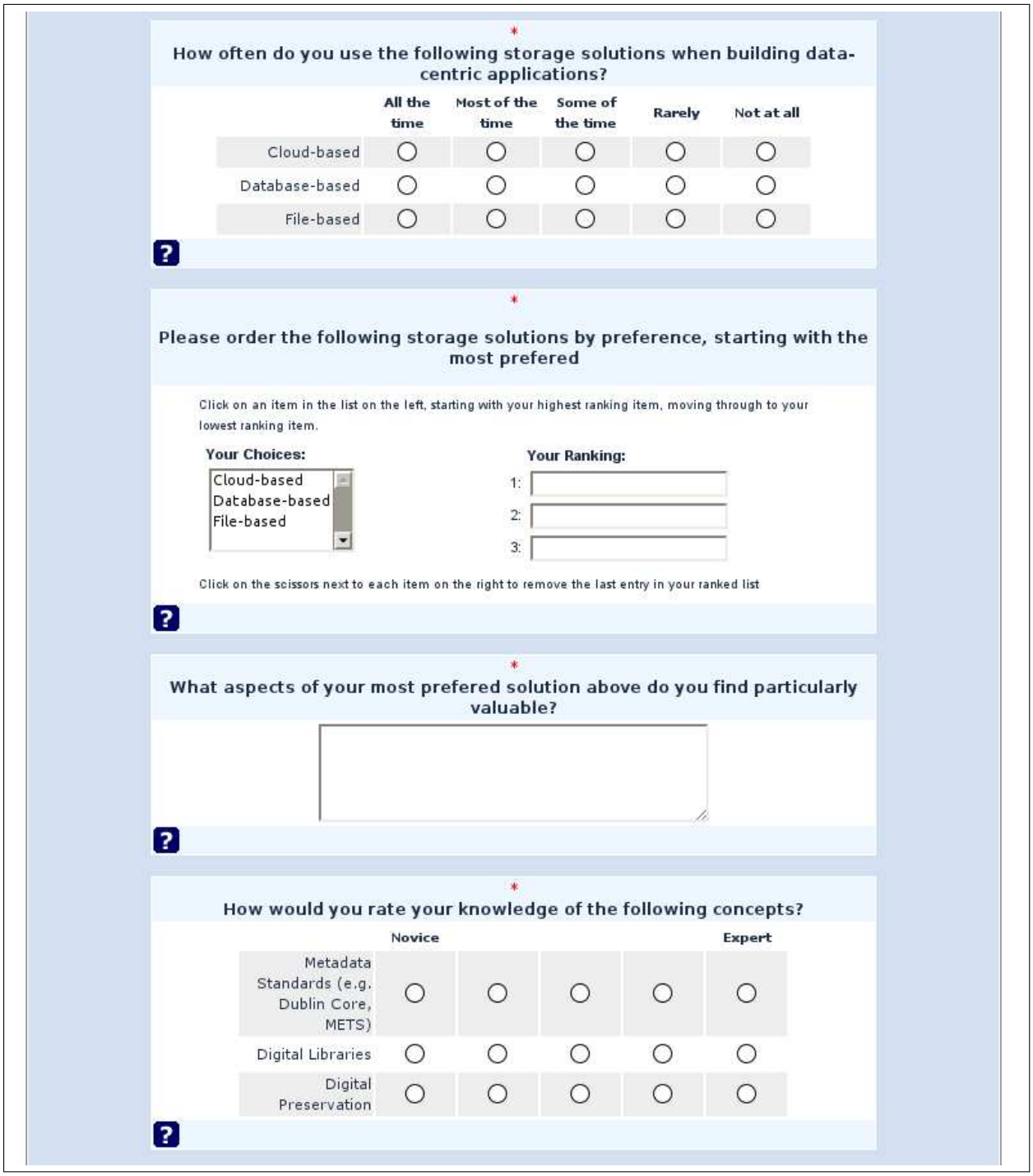
Please choose..

* **What is your experience working with the following tools and techniques?**

	Less than 1 year	1-3 years	3-6 years	More than 6 years
Web-based Application Development	○	○	○	○
Database-driven Applications	○	○	○	○
Database Management Systems (e.g. MySQL, MS Access, Oracle)	○	○	○	○
Extensible Markup Language (XML)	○	○	○	○

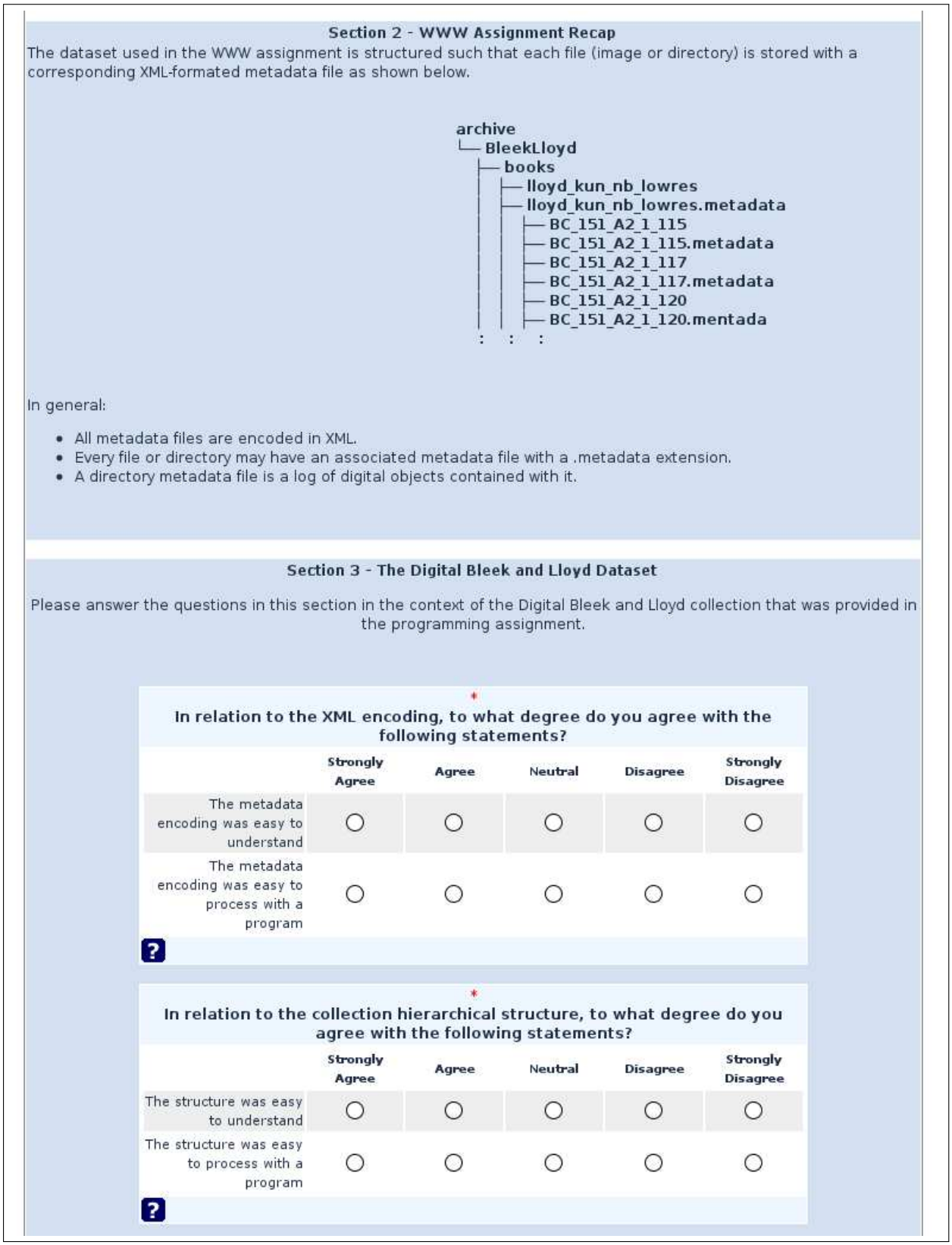
(a)

Figure A-5. Screenshot showing the online LimeSurvey questionnaire (page 1 of 5)



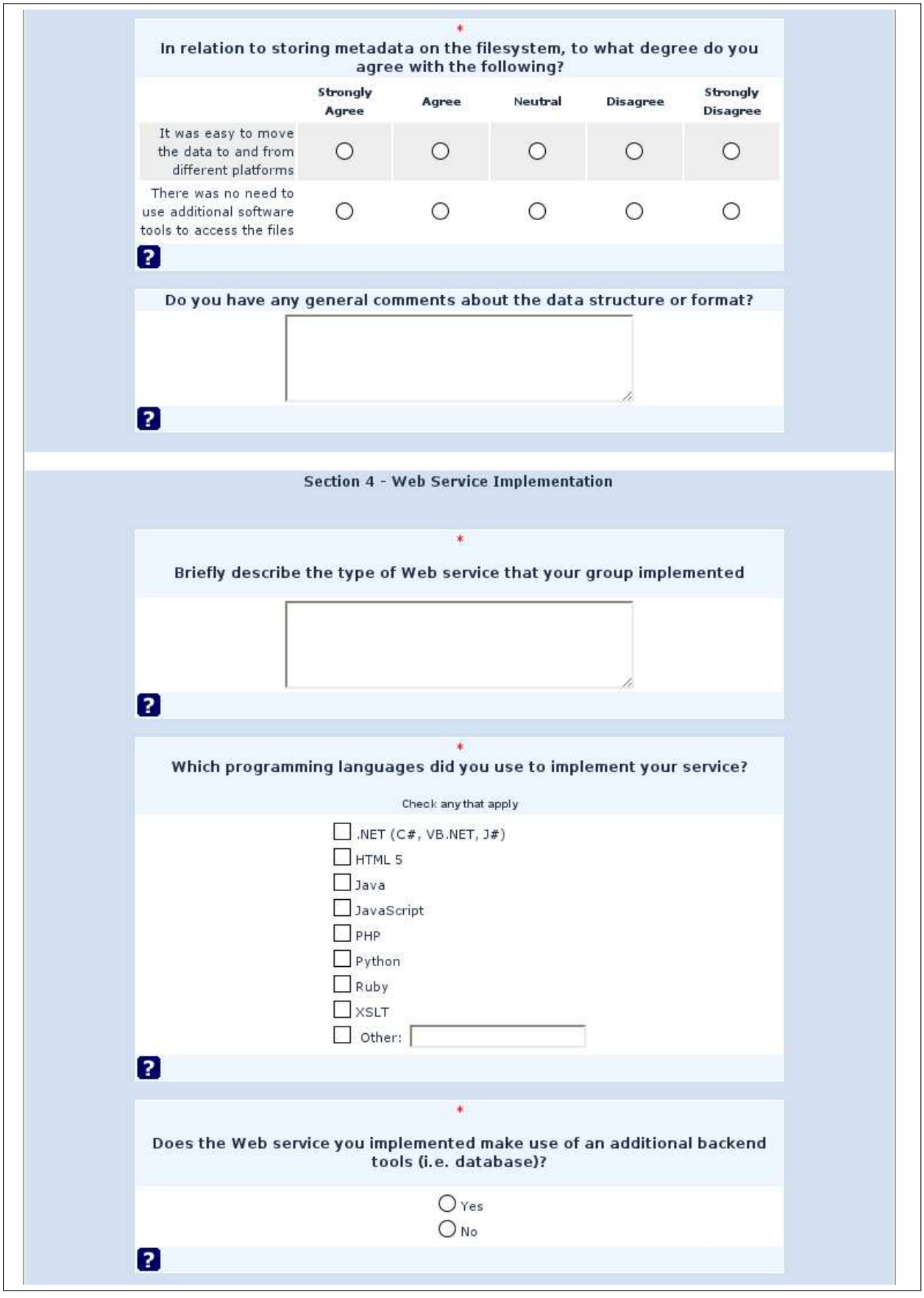
(b)

Figure A-5. Screenshot showing the online LimeSurvey questionnaire (page 2 of 5)



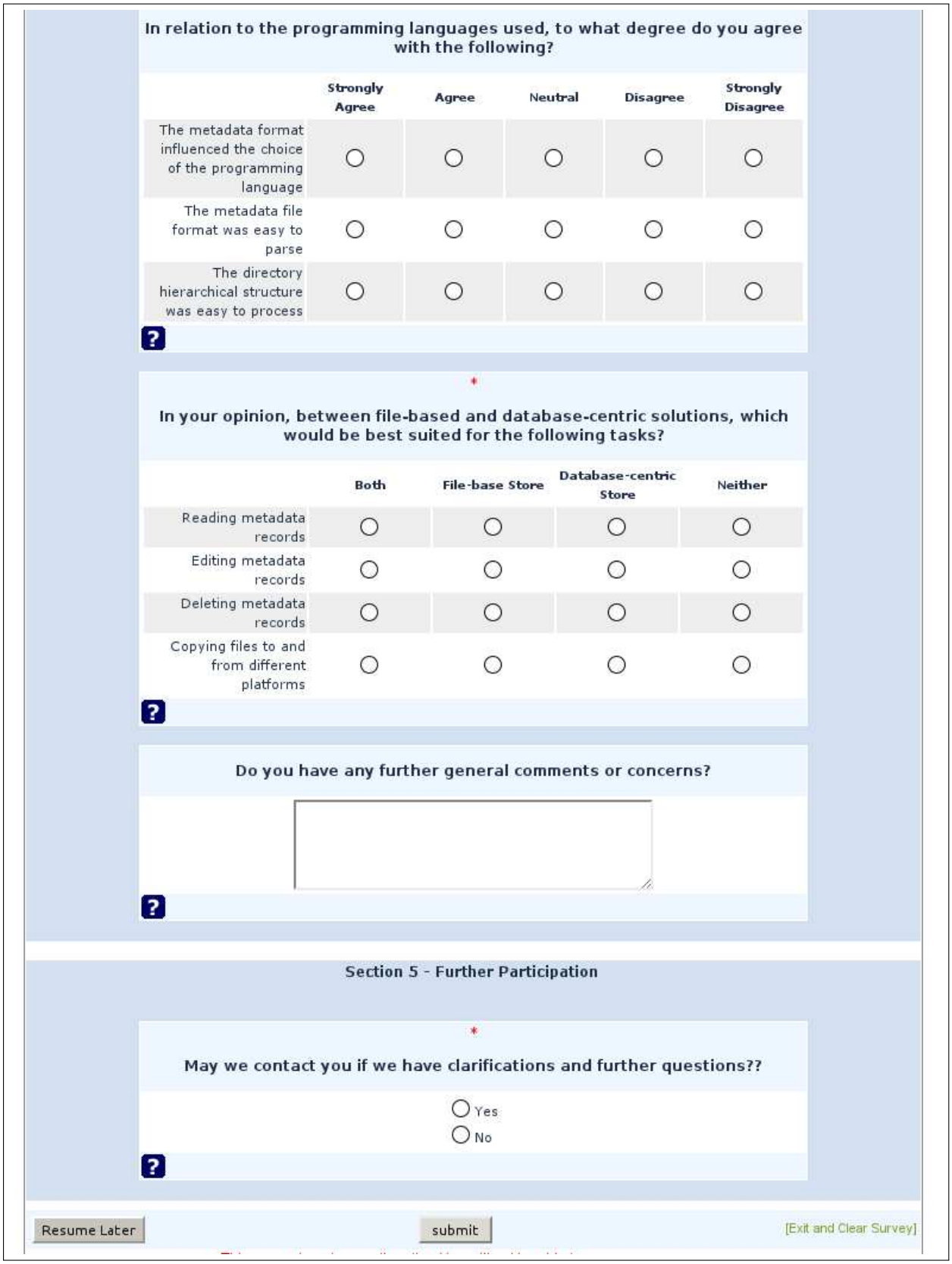
(c)

Figure A-5. Screenshot showing the online LimeSurvey questionnaire (page 3 of 5)



(d)

Figure A-5. Screenshot showing the online LimeSurvey questionnaire (page 4 of 5)



(e)

Figure A-5. Screenshot showing the online LimeSurvey questionnaire (page 5 of 5)

Appendix B

Experiment raw data

B.1 Developer Survey results

Table B-1. Developer survey raw data for technologies background

Experience working with DL tools and techniques					
	1 year ∨	**1-3 years**	**3-6 years**	**> 6 years**	
[Database Management Systems]					
	5	19	1	1	
[Database-Driven Applications]					
	13	11	1	1	
[Extensible Markup Language]					
	12	13	1	0	
[Web-Based Application Development]					
	14	10	1	1	

Table B-2. Developer survey raw data for DL concepts background

Participants knowledge of DL concepts						
	Novice				Expert	
	1	**2**	**3**	**4**	**5**	
[Digital Libraries]						
	10	11	5	0	0	
[Digital Preservation]						
	11	8	7	0	0	
[Metadata Standards]						
	8	9	8	1	0	

Table B-3. Developer survey raw data for storage usage frequencies

Storage solutions usage frequencies						
	All the time	**Most times**	**Not at all**	**Rarely**	**Some times**	
[Cloud-Based Solutions]						
	0	0	13	8	5	
[Database-Based Solutions]						
	4	7		5	10	
[File-Based Solutions]						
	6	6	1	5	8	

Table B-4. Developer survey raw data for storage rankings

Storage solutions preferences				
	Cloud	**Database**	**File**	
[Ranking 1]				
	8	12	6	
[Ranking 2]				
	5	10	11	
[Ranking 3]				
	13	4	9	
Reasons for most prefered solution				

Table B-5. Developer survey raw data for repository structure

To what degree do you agree with the following						
	Strong Agree	**Agree**	**Neutral**	**Disagree**	**Strong Disagree**	
[Easy to move the data]						
	3	15	6	2	0	
[No additional softwar required]						
	5	13	6	2	0	
[Easy to process with program]						
	5	11	4	5	1	
[Easy to understand]						
	2	10	8	6	0	
[XML was easy to process]						
	6	13	1	5	1	
[XML was easy to understand]						

(Continued on next page)

Table B-5. (continued)

To what degree do you agree with the following						
	Strong Agree	**Agree**	**Neutral**	**Disagree**	**Strong Disagree**	
	4	12	7	3	0	

Table B-6. Developer survey raw data for data management options

Solution best suited for data operations					
	Both	**Database**	**File Store**	**Neither**	
[Copying files]					
	1	5	19	1	
[Deleting metadata records]					
	6	11	9	0	
[Editing metadata records]					
	6	8	12	0	
[Reading metadata records]					
	5	7	13	1	

Table B-7. Developer survey raw data for programming languages

Programming languages used during in assignment							
	C#	**HTML5**	**Java**	**JavaScript**	**PHP**	**Python**	
	2	11	2	17	15	4	

Table B-8. Developer survey raw data for additional backend tools

Additional backend tools used in assignment			
	Yes	**No**	
	11	15	

Table B-9. Developer survey raw data for programming languages

To what degree do you agree with the following						
	Strong Agree	**Agree**	**Neutral**	**Disagree**	**Strong Disagree**	
[structure was easy to process]						
	11	3	7	5	0	
[metadata was easy to parse]						
	16	5	3	2	0	
[metadata influenced language]						
	3	6	11	1	5	

B.2 Performance benchmarks results

B.2.1 Workload models

Table B-10. Performance experiment raw data for dataset models

		Size	**1**	**2**	**3**	Σ
1						
	W1	0.53	19	—	—	19
	W2	0.97	25	—	—	25
	W3	2	42	—	—	42
	W4	3.9	57	—	—	57
	W5	7.6	67	—	—	67
	W6	15	83	—	—	83
	W7	30	100	—	—	100
	W8	60	112	—	—	112
	W9	118	116	—	—	116
	W10	236	119	—	—	119
	W11	471	127	—	—	127
	W12	942	129	—	—	129
	W13	1945.6	128	—	—	128
	W14	3788.8	131	—	—	131
	W15	7577.6	131	—	—	131
2						
	W1	0.78	19	66	—	85
	W2	1.4	25	105	—	130
	W3	2.7	42	186	—	228
	W4	4.9	57	264	—	321
	W5	9.2	67	420	—	487
	W6	17	83	551	—	634
	W7	33	100	771	—	871
	W8	64	112	1071	—	1183
	W9	123	116	1314	—	1430

(Continued on next page)

Table B-10. (continued)

		Size	1	2	3	Σ
	W10	243	119	1687	1	1807
	W11	481	127	2058	—	2185
	W12	957	129	2400	—	2529
	W13	1945.6	128	2747	—	2875
	W14	3891.2	131	3093	1	3225
	W15	7680	131	3457	1	3589
3						
	W1	1.2	19	66	96	181
	W2	2.1	25	105	174	304
	W3	4	42	186	331	559
	W4	7.2	57	264	585	906
	W5	14	67	420	1035	1522
	W6	24	83	551	1736	2370
	W7	44	100	771	2854	3725
	W8	80	112	1071	4499	5682
	W9	147	116	1314	6612	8042
	W10	277	119	1687	9689	11 495
	W11	526	127	2059	13 335	15 521
	W12	1016	129	2401	18 012	20 542
	W13	2048	128	2748	23 664	26 540
	W14	3993.6	131	3094	30 177	33 402
	W15	7782.4	131	3460	37 357	40 948

B.2.2 Ingestion

Table B-11. Performance experiment raw data for ingestion

		1	2	3	4	5
1	**Phase = Overall**					
	W1	289.64	2.83	10.29	2.76	3.28
	W2	53.66	2.79	2.91	14.28	2.8
	W3	47.44	2.77	2.77	2.84	2.77
	W4	34.32	2.77	2.76	2.79	2.81
	W5	27.63	2.79	2.92	2.78	2.87
	W6	1418.27	2.76	2.78	2.8	2.76
	W7	48.34	4.52	2.9	2.92	2.99
	W8	60.58	2.78	2.8	2.76	2.85
	W9	76.6	2.78	2.78	2.76	2.86
	W10	247.31	2.99	2.84	2.87	2.85
	W11	41.79	2.97	3.05	3	2.8
	W12	64.93	2.9	5.5	4.53	2.86
	W13	52.42	3.04	3.08	2.77	2.77
	W14	33	2.81	2.76	2.97	2.77
	W15	56.96	2.79	2.81	2.93	3.21
1	**Phase = Parsing**					
	W1	242.19	2.67	3.63	2.64	3.16
	W2	2.7	2.67	2.78	14.14	2.68
	W3	2.71	2.65	2.65	2.72	2.65

(Continued on next page)

Table B-11. (continued)

		1	2	3	4	5
	W4	2.65	2.65	2.64	2.67	2.69
	W5	2.66	2.64	2.79	2.66	2.75
	W6	2.65	2.64	2.66	2.68	2.64
	W7	2.7	4.39	2.78	2.79	2.84
	W8	2.68	2.66	2.67	2.64	2.73
	W9	2.68	2.66	2.66	2.64	2.74
	W10	2.69	2.87	2.72	2.72	2.72
	W11	3.04	2.82	2.91	2.85	2.68
	W12	2.7	2.77	5.33	4.39	2.71
	W13	2.72	2.9	2.92	2.65	2.65
	W14	2.67	2.68	2.64	2.84	2.64
	W15	2.83	2.66	2.68	2.8	3.06
	Phase = Disk Write					
	W1	47.44	0.16	6.66	0.12	0.12
	W2	50.96	0.12	0.14	0.13	0.12
	W3	44.73	0.12	0.12	0.12	0.12
	W4	31.66	0.12	0.12	0.12	0.12
	W5	24.97	0.15	0.13	0.12	0.13
	W6	1415.62	0.12	0.12	0.12	0.12
	W7	45.64	0.12	0.12	0.14	0.14
	W8	57.9	0.12	0.13	0.12	0.13
	W9	73.92	0.12	0.12	0.12	0.12
	W10	244.61	0.12	0.12	0.14	0.13
	W11	38.75	0.15	0.14	0.14	0.13
	W12	62.22	0.13	0.18	0.14	0.15
	W13	49.7	0.15	0.16	0.12	0.13
	W14	30.34	0.13	0.12	0.13	0.13
	W15	54.13	0.13	0.13	0.13	0.15
2	**Phase = Overall**					
	W1	98.12	2.97	2.94	7.3	3.38
	W2	90.01	2.82	11.69	3	2.85
	W3	38.52	2.84	2.78	3.02	2.83
	W4	75.7	2.77	7.53	3	3.04
	W5	43.22	2.77	3.12	2.76	2.8
	W6	52.64	2.78	2.75	3.24	2.82
	W7	83.54	2.79	2.78	2.84	2.84
	W8	52.18	2.79	2.79	2.77	2.85
	W9	1652.69	2.79	2.76	2.97	2.9
	W10	231.19	2.77	2.8	2.81	2.8
	W11	74.39	2.82	2.82	2.78	2.83
	W12	84.57	3.09	2.82	2.83	3.09
	W13	121.48	4.02	2.92	2.79	2.79
	W14	108.62	2.74	2.81	2.96	2.9
	W15	77.69	3.14	2.77	3.1	2.8
2	**Phase = Parsing**					
	W1	2.7	2.83	2.74	7.17	3.26
	W2	2.71	2.7	11.54	2.87	2.72
	W3	2.89	2.72	2.66	2.87	2.71
	W4	2.77	2.65	7.39	2.86	2.9
	W5	2.76	2.65	2.97	2.64	2.68
	W6	2.68	2.66	2.64	3.1	2.7
	W7	2.67	2.68	2.67	2.72	2.72

(Continued on next page)

Table B-11. (continued)

		1	2	3	4	5
	W8	2.99	2.67	2.67	2.65	2.72
	W9	2.64	2.67	2.64	2.79	2.78
	W10	2.66	2.65	2.68	2.69	2.68
	W11	2.68	2.7	2.65	2.66	2.71
	W12	2.97	2.94	2.7	2.7	2.95
	W13	2.79	3.89	2.8	2.67	2.67
	W14	2.73	2.62	2.69	2.83	2.77
	W15	2.66	2.98	2.64	2.97	2.67
	Phase = Disk Write					
	W1	95.42	0.14	0.2	0.14	0.13
	W2	87.29	0.12	0.15	0.12	0.13
	W3	35.64	0.12	0.13	0.15	0.12
	W4	72.93	0.12	0.14	0.14	0.13
	W5	40.46	0.12	0.15	0.12	0.12
	W6	49.96	0.12	0.12	0.14	0.12
	W7	80.88	0.12	0.12	0.12	0.12
	W8	49.18	0.12	0.12	0.12	0.13
	W9	1650.05	0.12	0.12	0.18	0.12
	W10	228.53	0.12	0.12	0.12	0.12
	W11	71.71	0.12	0.16	0.12	0.12
	W12	81.6	0.15	0.12	0.13	0.14
	W13	118.68	0.13	0.13	0.13	0.12
	W14	105.88	0.12	0.12	0.13	0.12
	W15	75.02	0.15	0.12	0.13	0.13
3	**Phase = Overall**					
	W1	100.79	3.03	3.2	7.43	2.83
	W2	48.85	6.8	3	2.78	3.51
	W3	62.4	2.78	2.81	3.14	3.05
	W4	67.49	2.86	2.86	2.86	2.77
	W5	32.98	3.03	2.92	2.84	2.96
	W6	63.31	2.8	2.8	3.05	2.78
	W7	93.01	2.93	2.76	3.07	2.82
	W8	62.29	2.85	2.93	2.87	2.79
	W9	67.74	2.8	3.03	3	10.96
	W10	81.61	3.06	2.79	2.82	2.84
	W11	75.08	2.97	2.78	2.86	3.17
	W12	108.41	2.76	3	2.98	2.75
	W13	358.58	2.77	2.75	2.77	2.94
	W14	80.5	2.76	2.82	2.78	2.75
	W15	106.17	2.79	2.85	2.85	2.79
3	**Phase = Parsing**					
	W1	13.55	2.9	3.05	7.3	2.69
	W2	3.24	6.6	2.86	2.66	3.39
	W3	2.65	2.65	2.69	2.99	2.91
	W4	2.72	2.73	2.74	2.74	2.65
	W5	2.7	2.89	2.79	2.72	2.83
	W6	2.86	2.67	2.68	2.9	2.65
	W7	2.66	2.72	2.64	2.94	2.7
	W8	2.66	2.68	2.76	2.73	2.68
	W9	2.66	2.68	2.89	2.85	2.82
	W10	2.67	2.92	2.67	2.7	2.71
	W11	2.7	2.82	2.66	2.74	3.03

(Continued on next page)

Table B-11. (continued)

		1	2	3	4	5
	W12	2.86	2.64	2.86	2.85	2.63
	W13	2.78	2.65	2.63	2.65	2.82
	W14	2.67	2.64	2.7	2.66	2.63
	W15	2.67	2.67	2.73	2.73	2.64
	Phase = Disk Write					
	W1	87.24	0.14	0.14	0.13	0.14
	W2	45.61	0.2	0.14	0.12	0.12
	W3	59.75	0.12	0.12	0.15	0.14
	W4	64.78	0.12	0.12	0.13	0.12
	W5	30.28	0.14	0.13	0.12	0.13
	W6	60.45	0.13	0.12	0.15	0.12
	W7	90.34	0.21	0.12	0.13	0.12
	W8	59.64	0.17	0.17	0.15	0.12
	W9	65.09	0.12	0.14	0.14	8.14
	W10	78.94	0.14	0.12	0.12	0.13
	W11	72.39	0.15	0.12	0.12	0.15
	W12	105.54	0.12	0.14	0.13	0.12
	W13	355.8	0.12	0.12	0.12	0.12
	W14	77.83	0.12	0.12	0.12	0.12
	W15	103.5	0.12	0.12	0.12	0.14

B.2.3 Search

Table B-12. Performance experiment raw data for search

		1	2	3	4	5
1	**Phase = Overall**					
	W1	195.18	24.62	24.63	24.64	24.77
	W2	282.11	47.93	47.94	47.63	47.99
	W3	404.63	97.67	97.15	97.24	97.48
	W4	1170.46	191.64	191.56	192.26	192.14
	W5	2677.09	387.17	387.27	386.95	386.57
	W6	2936.71	768.24	768.21	771.62	765.32
	W7	6624.93	1544.51	1524.79	1529.03	1525.89
	W8	43 226	3072.92	3059.75	3116.66	3123.16
	W9	137 444.38	6239.22	6154.6	6144.61	6150.45
	W10	318 844.87	12 403.6	12 422.68	12 710.26	12 411.71
	W11	780 773.27	25 409.49	25 141.06	25 175.92	24 708.49
	W12	1 622 021.39	49 853.95	49 048.52	49 161.42	49 141.91
	W13	3 875 299.14	104 027.86	99 129.04	99 040.29	98 872.15
	W14	8 933 491.31	7 335 941.65	7 416 207.84	7 350 024.06	7 358 842.46
	W15	21 932 576.11	18 306 767.28	18 408 131.83	19 536 419.13	18 407 536.35
1	**Phase = Parsing**					
	W1	79.75	9.51	9.44	9.47	9.56
	W2	147.72	18.79	18.69	18.54	18.77
	W3	224.98	38.35	37.76	37.83	38.02
	W4	582.86	75.7	75.76	75.73	75.61
	W5	1305.56	153.27	152.39	151.77	152.33

(Continued on next page)

Table B-12. (continued)

		1	2	3	4	5
	W6	1791.52	304.51	304.46	305.47	301.87
	W7	4407.16	606.79	601.31	602.4	600.66
	W8	38 438	1216.69	1204.56	1233.3	1234.78
	W9	131 226.89	2486.87	2447.74	2441.93	2440.03
	W10	306 332.31	4960.09	4942.19	4976	4961
	W11	757 748.29	9989.67	9934.02	9890.42	9848.33
	W12	1 558 213.23	19 602.24	19 520.14	19 575.47	19 487.99
	W13	3 721 310.45	40 668.23	39 724.13	39 670.54	39 775.52
	W14	8 552 306.79	7 021 045.27	7 085 465.93	7 019 735.31	7 030 185.89
	W15	20 686 750.58	17 008 969.14	17 081 428.81	18 163 715.62	17 084 730.98
	Phase = XPath					
	W1	21.87	11.12	11.21	11.21	11.23
	W2	45.57	21.77	21.85	21.77	21.82
	W3	95.19	44.72	44.94	44.9	45.01
	W4	185.89	88.64	88.49	89.1	88.96
	W5	374.74	181.03	181.19	181.41	180.6
	W6	756.94	360.08	359.64	360.86	359.12
	W7	1510.18	728.33	716.87	720.85	717.06
	W8	2984.49	1444.86	1444.16	1464.8	1470.6
	W9	4194.45	2916.37	2884.35	2879.51	2890.78
	W10	8128.19	5798.03	5826.67	5850.69	5813.13
	W11	15 524.59	11 644.32	11 667.39	11 615.07	11 588.4
	W12	43 710.53	23 137.13	23 082.77	23 058	23 104.57
	W13	79 438.77	46 074.33	46 116.43	46 116.66	45 968.05
	W14	143 938.7	179 589.99	179 913.71	179 929.56	179 977.18
	W15	280 066.58	354 841.74	353 939.2	314 975.65	356 153.99
	Phase = Traversal					
	W1	93.56	3.99	3.99	3.96	3.98
	W2	88.82	7.37	7.41	7.32	7.4
	W3	84.47	14.6	14.45	14.51	14.44
	W4	401.72	27.3	27.32	27.43	27.57
	W5	996.79	52.87	53.68	53.77	53.64
	W6	388.25	103.65	104.1	105.29	104.33
	W7	707.59	209.4	206.61	205.78	208.17
	W8	1803.51	411.36	411.03	418.56	417.78
	W9	2023.03	835.97	822.51	823.17	819.63
	W10	4384.37	1645.49	1653.81	1883.57	1637.59
	W11	7500.4	3775.5	3539.65	3670.43	3271.76
	W12	20 097.64	7114.59	6445.61	6527.95	6549.34
	W13	74 549.92	17 285.3	13 288.48	13 253.08	13 128.59
	W14	237 245.82	135 306.39	150 828.19	150 359.2	148 679.4
	W15	965 758.95	942 956.4	972 763.82	1 057 727.86	966 651.38
2	**Phase = Overall**					
	W1	256.83	27.3	27.22	27.21	27.18
	W2	1611.77	52.18	51.75	51.94	51.77
	W3	1339.74	106.69	105.88	105.45	105.49
	W4	1808.64	204.84	204.56	203.99	202.77
	W5	3902.07	406.19	406.43	405.94	406.79
	W6	6101.84	799.44	792.39	793.53	795.93
	W7	10 027.94	1575.02	1572.9	1579.31	1569.4
	W8	18 524.15	3134.83	3136.95	3132.46	3128.85
	W9	33 029.93	6229.91	6218.97	6202.27	6241.36

(Continued on next page)

Table B-12. (continued)

		1	2	3	4	5
	W10	89 834.7	12 850.96	12 395.74	12 432.91	12 395.91
	W11	198 638.61	25 524.99	24 809.76	24 764.96	24 832.72
	W12	659 394.2	52 176.44	49 558.25	49 441.72	49 273.51
	W13	1 371 705.38	107 911.19	99 960.16	99 375.28	99 734.05
	W14	3 790 555.62	3 454 981.67	3 461 207.67	3 455 124.86	3 464 561.81
	W15	11 704 076.22	10 700 810.89	11 717 518.55	10 969 375.79	10 694 511.63
2				**Phase = Parsing**		
	W1	64.78	9.59	9.52	9.5	9.45
	W2	407.09	18.71	18.43	18.48	18.42
	W3	558.3	38.49	38.34	38.03	38.09
	W4	720.49	76.97	75.75	75.82	75.41
	W5	1827.85	154.26	154.03	153.12	153.35
	W6	3245.83	304.63	305.04	305.53	302.84
	W7	5586.71	613.66	612.4	617.38	608.2
	W8	10 054.6	1231.14	1226.16	1228.44	1223.14
	W9	19 059.3	2465.62	2446.35	2448.87	2465.2
	W10	62 362.28	4938.41	4896.42	4914.15	4915.12
	W11	151 842.75	9932.74	9866.18	9817.9	9844.75
	W12	575 706.01	19 957.37	19 776.84	19 678.71	19 606.73
	W13	1 219 740.23	40 128.09	39 998.58	39 794.58	39 929.06
	W14	3 485 885.31	3 164 139.22	3 170 847.95	3 163 687.76	3 173 643.79
	W15	11 122 506.05	10 116 372.78	11 224 523.28	10 404 928.25	10 113 103.48
				Phase = XPath		
	W1	22.58	11.23	11.21	11.23	11.25
	W2	44.86	21.91	21.83	21.98	21.89
	W3	92.3	45.63	45.64	45.66	45.59
	W4	187.4	89.64	90.4	90.03	89.53
	W5	372.79	181.44	181.39	181.76	182.28
	W6	751.2	365.47	359.51	360.5	365.22
	W7	1493.16	720.76	718.19	719.69	717.63
	W8	3025.15	1444.49	1448.88	1442.75	1446.39
	W9	6015.72	2883.11	2885.6	2876.77	2894.64
	W10	11 990.97	5756.07	5764.56	5789.81	5771.21
	W11	24 068.74	11 511.22	11 552.84	11 556.24	11 567.42
	W12	47 408.32	23 084.04	23 087.85	23 083.97	23 036.41
	W13	95 045.1	46 435.26	46 272.05	46 154.37	46 411.89
	W14	187 891.87	189 383.56	188 677.4	189 199	188 831.15
	W15	364 615.28	374 103.6	303 449.81	355 344.02	371 536.61
				Phase = Traversal		
	W1	169.48	6.48	6.49	6.47	6.49
	W2	1159.83	11.55	11.5	11.48	11.45
	W3	689.15	22.57	21.89	21.76	21.81
	W4	900.75	38.23	38.41	38.14	37.83
	W5	1701.42	70.49	71.01	71.06	71.16
	W6	2104.81	129.34	127.84	127.51	127.88
	W7	2948.07	240.6	242.31	242.24	243.57
	W8	5444.4	459.2	461.92	461.27	459.32
	W9	7954.9	881.18	887.02	876.63	881.52
	W10	15 481.45	2156.48	1734.76	1728.96	1709.59
	W11	22 727.13	4081.03	3390.75	3390.82	3420.55
	W12	36 279.87	9135.02	6693.56	6679.04	6630.37
	W13	56 920.04	21 347.84	13 689.52	13 426.33	13 393.1

(Continued on next page)

Table B-12. (continued)

		1	2	3	4	5
	W14	116 778.44	101 458.88	101 682.32	102 238.11	102 086.87
	W15	216 954.89	210 334.51	189 545.45	209 103.52	209 871.54
3	**Phase = Overall**					
	W1	298.72	30.51	30.43	30.78	30.3
	W2	1783.19	57.75	57.3	57.72	57.39
	W3	1755.77	116.65	115.88	115.68	116.26
	W4	3995.3	225.15	223.51	223.21	225.64
	W5	7020.17	443.2	440.42	441.21	440.47
	W6	13 871.57	861.75	858.38	860.7	858.09
	W7	23 016.81	2225.3	1672.35	1672.81	1672.15
	W8	48 331.7	3427.63	3303.07	3304.47	3308.03
	W9	70 237.16	6855.59	6484.46	6467.21	6485.01
	W10	68 207.45	14 348.67	12 816.83	12 805.15	12 806.8
	W11	188 772.37	29 646.99	25 382.97	25 427.95	25 524.33
	W12	399 812.99	64 790.77	50 586.88	50 537.15	50 532.63
	W13	692 391.6	183 192.13	102 811.77	100 788.36	100 336.3
	W14	1 449 066.95	1 355 233.19	1 367 962.81	1 366 005.77	1 359 123.36
	W15	3 377 263.84	3 133 229.84	3 120 101.94	3 123 345.38	3 124 346.77
3	**Phase = Parsing**					
	W1	50.01	9.52	9.45	9.49	9.38
	W2	369.59	18.44	18.22	18.32	18.33
	W3	648.88	38.08	37.48	37.49	37.64
	W4	1322.62	75.79	74.19	74.54	75.17
	W5	2599.35	151.77	150.2	150.33	150.37
	W6	5170.64	302.96	300.13	300.86	299.18
	W7	9530.53	610.75	602.49	600.83	596.33
	W8	19 576.72	1229.89	1212.82	1215.16	1215.3
	W9	31 131.14	2468.01	2437.99	2423.69	2432.96
	W10	32 769.04	5074.6	4895.8	4897.57	4887.42
	W11	90 253.66	10 094.84	9856.98	9874.35	10 010.76
	W12	211 274.05	20 576.71	19 861.06	19 851.95	19 885.1
	W13	413 092.3	43 381.81	39 936.26	39 796.42	39 657.27
	W14	1 000 349.19	922 768.17	931 480.33	930 961.2	923 140.54
	W15	2 590 911.01	2 366 361.88	2 353 697.22	2 357 864.22	2 359 454.83
	Phase = XPath					
	W1	24.17	11.39	11.35	11.58	11.4
	W2	45.38	21.96	21.97	21.91	21.91
	W3	92.16	45.69	45.6	45.55	45.73
	W4	184.61	90.69	90.89	90.35	92.09
	W5	370.85	183.13	182.3	182.94	182.38
	W6	745.6	367.76	365.38	367.72	367.79
	W7	1468.94	725.45	720.38	720.13	730.55
	W8	2956.34	1463.02	1452.72	1453	1454.57
	W9	5967.63	2914.94	2903.01	2899.05	2899.12
	W10	12 103.48	5956.06	5791.48	5784.85	5797.43
	W11	23 891.75	11 726.21	11 555.66	11 538.96	11 534.16
	W12	47 855.91	23 801.63	23 164.45	23 110.91	23 033.53
	W13	96 062.05	50 340.98	46 426.84	46 306.68	46 206.68
	W14	191 225.21	191 918.16	192 719.44	191 766.13	191 808.71
	W15	384 410.01	385 169.63	384 806.44	384 357.25	385 044.03
	Phase = Traversal					
	W1	224.55	9.6	9.63	9.71	9.52

(Continued on next page)

Table B-12. (continued)

		1	2	3	4	5
	W2	1368.22	17.34	17.11	17.49	17.15
	W3	1014.73	32.87	32.79	32.64	32.89
	W4	2488.07	58.68	58.44	58.31	58.39
	W5	4049.98	108.3	107.92	107.94	107.72
	W6	7955.34	191.03	192.87	192.12	191.12
	W7	12 017.35	889.1	349.49	351.85	345.27
	W8	25 798.63	734.72	637.53	636.31	638.16
	W9	33 138.39	1472.64	1143.45	1144.48	1152.93
	W10	23 334.93	3318	2129.55	2122.74	2121.95
	W11	74 626.97	7825.95	3970.33	4014.64	3979.41
	W12	140 683.04	20 412.43	7561.38	7574.28	7614.01
	W13	183 237.25	89 469.33	16 448.67	14 685.25	14 472.35
	W14	257 492.56	240 546.86	243 763.03	243 278.44	244 174.11
	W15	401 942.83	381 698.33	381 598.29	381 123.91	379 847.91

B.2.4 OAI-PMH data provider

Table B-13. Performance experiment raw data for OAI-PMH

		1	2	3	4	5
1	**Verb = GetRecord**					
	W1	0.01	0.01	0.01	0.01	0.01
	W2	0.01	0.01	0.01	0.01	0.01
	W3	0.02	0.02	0.02	0.01	0.02
	W4	0.02	0.02	0.02	0.02	0.02
	W5	0.04	0.02	0.03	0.03	0.03
	W6	0.05	0.05	0.05	0.05	0.05
	W7	0.08	0.08	0.08	0.09	0.08
	W8	0.16	0.16	0.16	0.16	0.15
	W9	0.3	0.3	0.3	0.31	0.3
	W10	0.61	0.6	0.6	0.59	0.6
	W11	1.2	1.21	1.2	1.22	1.2
	W12	2.47	2.46	2.46	2.45	2.47
	W13	4.9	4.92	4.93	4.94	4.87
	W14	22.81	21.11	10.86	10.72	10.82
	W15	608.6	673.77	611.09	662.52	608.11
1	**Verb = ListIdentifiers**					
	W1	0.01	0.01	0.02	0.01	0.01
	W2	0.03	0.03	0.03	0.02	0.03
	W3	0.06	0.07	0.06	0.06	0.06
	W4	0.17	0.17	0.17	0.18	0.17
	W5	0.4	0.39	0.39	0.4	0.4
	W6	0.77	0.77	0.77	0.77	0.77
	W7	1.53	1.54	1.53	1.53	1.54
	W8	3.09	3.09	3.09	3.09	3.1
	W9	6.25	6.34	6.37	6.37	6.35
	W10	13.38	13.51	13.47	13.48	13.68
	W11	28.46	29.09	29.21	29.67	29.68

(Continued on next page)

Table B-13. (continued)

		1	2	3	4	5
	W12	64.22	61.86	64.02	64.41	65.6
	W13	140.49	133.94	138.99	141.51	143.79
	W14	548.08	566.62	567.82	580.47	589.56
	W15	1922.89	2035.21	1981.42	2050.59	2081.8
	Verb = ListRecords					
	W1	1.93	0.52	0.49	0.48	0.5
	W2	1.11	0.98	0.98	0.99	0.98
	W3	2.25	2	2.01	2.01	2.03
	W4	6.7	4.12	4.23	4.23	4.22
	W5	6.64	5.41	5.54	5.44	5.53
	W6	6.55	5.94	5.95	5.95	5.96
	W7	7.48	6.73	6.75	6.79	6.76
	W8	9.81	8.37	8.45	8.52	8.45
	W9	13.45	11.68	12.07	11.93	12.16
	W10	20.88	18.9	18.96	19.33	19.23
	W11	38.09	33.46	34.95	34.39	35.5
	W12	81.16	66.95	69.08	67.75	68.89
	W13	210.85	139.72	141.5	146.33	145.58
	W14	590.57	590.64	586.45	602.66	615.35
	W15	1991.68	1972.84	1937.09	2029.23	2011.34
1	**Verb = ListSets**					
	W1	0.01	0	0.01	0.01	0.01
	W2	0.01	0.01	0.01	0.01	0.01
	W3	0.01	0.01	0.01	0.02	0
	W4	0.01	0.02	0.02	0.01	0.01
	W5	0.02	0.02	0.02	0.01	0.01
	W6	0.03	0.03	0.02	0.03	0.03
	W7	0.05	0.04	0.05	0.04	0.04
	W8	0.07	0.07	0.07	0.08	0.07
	W9	0.13	0.12	0.14	0.13	0.13
	W10	0.26	0.25	0.25	0.25	0.24
	W11	0.5	0.49	0.5	0.5	0.49
	W12	1	0.99	1.01	1	1
	W13	2.05	1.99	1.99	2.03	2.02
	W14	119.78	116.76	119.58	75.76	11.61
	W15	628.31	658.13	638.11	609.18	647.21
2	**Verb = GetRecord**					
	W1	0.01	0.01	0.02	0.01	0.01
	W2	0.01	0.02	0.01	0.02	0.01
	W3	0.02	0.02	0.02	0.02	0.02
	W4	0.03	0.03	0.02	0.02	0.03
	W5	0.04	0.05	0.04	0.04	0.04
	W6	0.07	0.06	0.07	0.07	0.06
	W7	0.11	0.12	0.1	0.1	0.1
	W8	0.19	0.2	0.2	0.19	0.2
	W9	0.35	0.36	0.35	0.36	0.35
	W10	0.68	0.68	0.68	0.68	0.68
	W11	1.32	1.33	1.32	1.33	1.32
	W12	2.66	2.65	2.66	2.63	2.64
	W13	5.2	5.24	5.25	5.28	5.24
	W14	58.47	57.4	59.49	56.65	57.59
	W15	128.98	128	128.28	129.4	130.34

(Continued on next page)

Table B-13. (continued)

		1	2	3	4	5
2	**Verb = ListIdentifiers**					
	W1	0.02	0.03	0.02	0.03	0.01
	W2	0.03	0.03	0.03	0.04	0.03
	W3	0.07	0.07	0.07	0.07	0.07
	W4	0.2	0.18	0.19	0.19	0.19
	W5	0.41	0.42	0.42	0.42	0.42
	W6	0.8	0.8	0.8	0.81	0.81
	W7	1.57	1.59	1.59	1.58	1.59
	W8	3.16	3.21	3.18	3.24	3.19
	W9	6.51	6.47	6.64	6.53	6.64
	W10	13.45	13.82	13.68	14.14	13.61
	W11	29.37	29.98	29.87	30.2	30.84
	W12	64.93	66.66	67.39	67.9	66.02
	W13	141.48	142.09	143.05	146.62	146.72
	W14	432.59	470.88	435.17	457.24	452.32
	W15	973.71	1052.02	1049.53	1193.01	1130.72
	Verb = ListRecords					
	W1	1.94	0.55	0.49	0.49	0.49
	W2	2.41	0.99	1	0.99	0.99
	W3	3.23	2.01	2.03	2.03	2.03
	W4	5.89	4.17	4.25	4.27	4.27
	W5	7.95	5.47	5.51	5.51	5.56
	W6	9.34	5.99	5.97	5.99	5.99
	W7	10.64	6.8	6.81	6.78	6.82
	W8	12.03	8.49	8.59	8.62	8.65
	W9	18.16	11.97	12.28	12.2	12.29
	W10	29.66	19.1	19.63	19.48	19.94
	W11	48.69	34.63	36.19	35.24	35.68
	W12	90.59	69.82	71.94	72.58	72.27
	W13	185.46	147.83	152.61	148.73	156.54
	W14	445.35	425.54	428.54	453.39	471.71
	W15	1005.01	954.68	977.41	951.68	1060.94
2	**Verb = ListSets**					
	W1	0.02	0.01	0.01	0.01	0.01
	W2	0.01	0.02	0.01	0.01	0.02
	W3	0.02	0.03	0.02	0.01	0.02
	W4	0.03	0.02	0.03	0.02	0.03
	W5	0.04	0.04	0.04	0.03	0.04
	W6	0.06	0.05	0.06	0.04	0.06
	W7	0.08	0.08	0.08	0.08	0.08
	W8	0.13	0.12	0.12	0.13	0.12
	W9	0.21	0.2	0.19	0.2	0.2
	W10	0.36	0.34	0.34	0.35	0.35
	W11	0.64	0.62	0.62	0.62	0.63
	W12	1.2	1.19	1.2	1.19	1.2
	W13	2.29	2.28	2.28	2.28	2.29
	W14	53.84	54.22	55.45	55.19	53.11
	W15	121.05	121.63	125.47	126.05	125.03
3	**Verb = GetRecord**					
	W1	0.02	0.02	0.02	0.01	0.01
	W2	0.02	0.02	0.02	0.02	0.02
	W3	0.03	0.02	0.03	0.03	0.02

(Continued on next page)

Table B-13. (continued)

		1	2	3	4	5
	W4	0.05	0.04	0.05	0.04	0.04
	W5	0.07	0.07	0.06	0.08	0.07
	W6	0.11	0.11	0.1	0.11	0.11
	W7	0.19	0.19	0.19	0.19	0.19
	W8	0.32	0.33	0.32	0.31	0.33
	W9	0.55	0.54	0.55	0.54	0.54
	W10	0.97	0.98	1	0.97	0.99
	W11	1.75	1.73	1.77	1.75	1.75
	W12	3.18	3.2	3.24	3.23	3.19
	W13	6.18	6.16	6.28	6.23	6.2
	W14	169.94	168.91	172.72	170.87	169.65
	W15	258.1	257.12	257.81	258.33	257.43
3	**Verb = ListIdentifiers**					
	W1	0.02	0.03	0.03	0.02	0.02
	W2	0.04	0.05	0.04	0.04	0.04
	W3	0.09	0.08	0.08	0.09	0.08
	W4	0.21	0.22	0.22	0.21	0.22
	W5	0.47	0.47	0.47	0.47	0.47
	W6	0.89	0.9	0.89	0.9	0.9
	W7	1.74	1.75	1.74	1.76	1.75
	W8	3.44	3.52	3.47	3.55	3.47
	W9	6.9	7.25	6.93	7.33	7.02
	W10	14.66	15.93	15.26	15.54	14.23
	W11	28.77	32.92	29.26	32.33	30.62
	W12	56.51	68.71	58.32	70.74	61.98
	W13	141.94	155.22	143.46	157.94	148.13
	W14	699.98	704.1	716.51	751.32	736.28
	W15	1344.34	1342.29	1422.78	1417.14	1419.45
	Verb = ListRecords					
	W1	1.76	0.55	0.49	0.5	0.49
	W2	2.44	0.99	1	1	1.01
	W3	3.8	2.02	2.06	2.05	2.05
	W4	7.5	4.2	4.3	4.28	4.3
	W5	9.68	5.57	5.62	5.57	5.6
	W6	13.79	6.06	6.13	6.12	6.08
	W7	17.99	6.98	7	7.02	7.06
	W8	28.05	8.79	9.01	8.93	9.06
	W9	34.68	12.43	13.07	12.66	13.17
	W10	35.63	19.43	21.16	19.63	21.2
	W11	51.59	33.31	40.13	34.63	39.33
	W12	327.16	90.23	79.92	65.78	79.21
	W13	345.9	147.23	162.49	149.55	168.86
	W14	704.08	708.5	732.86	717.09	707.86
	W15	1318.5	1349.22	1395.19	1317.7	1398.06
3	**Verb = ListSets**					
	W1	0.02	0.02	0.02	0.01	0.02
	W2	0.02	0.03	0.02	0.02	0.03
	W3	0.03	0.04	0.04	0.03	0.04
	W4	0.06	0.05	0.06	0.06	0.05
	W5	0.09	0.1	0.08	0.09	0.09
	W6	0.15	0.14	0.14	0.14	0.14
	W7	0.24	0.23	0.22	0.23	0.22

(Continued on next page)

Table B-13. (continued)

		1	2	3	4	5
	W8	0.39	0.35	0.37	0.36	0.36
	W9	0.58	0.56	0.54	0.56	0.55
	W10	0.93	0.89	0.89	0.89	0.9
	W11	1.51	1.44	1.4	1.4	1.43
	W12	2.37	2.37	2.25	2.24	2.25
	W13	3.97	3.89	3.79	3.8	3.79
	W14	164.09	163.76	161.98	162.7	163.01
	W15	254.31	248.24	248.58	249.03	255.68

B.2.5 RSS feed generator

Table B-14. Performance experiment raw data for feed generator

		1	2	3	4	5
1	**Feed Size = 10**					
	W1	126.19	3.17	3.18	3.14	3.17
	W2	100.62	5.74	5.77	5.88	5.79
	W3	43.61	11.13	11.2	11.42	11.14
	W4	469.96	21.24	21.53	21.36	21.26
	W5	859.29	41.32	41.15	40.88	41.21
	W6	313.74	80.74	80.42	79.69	79.77
	W7	506.46	159.59	159.67	160.05	158.8
	W8	1231.32	312.62	314.24	314.61	309.86
	W9	1515.71	616.52	620.69	622.15	617.45
	W10	2910.32	1244.34	1228.22	1256.9	1238.55
	W11	5870.46	2472.77	2492.03	2493.73	2495.1
	W12	14 255.93	4968.46	4974.29	4977.62	4931.97
	W13	53 507.65	11 726.26	10 422.47	10 124.06	10 226.86
	W14	202 099.24	142 956.68	161 814.93	137 669.03	161 639.18
	W15	1 494 247.27	1 500 161.56	1 469 962.57	1 501 713.71	1 502 793.11
1	**Feed Size = 20**					
	W1	138.92	3.28	3.19	3.24	3.23
	W2	77.42	5.88	5.91	5.89	5.83
	W3	27.88	11.24	11.4	11.38	11.32
	W4	357.03	21.63	21.56	21.26	21.36
	W5	857.77	40.9	41.26	41.06	41.08
	W6	240.27	80.14	81.29	80.96	80.29
	W7	386.42	160	156.6	155.23	157.66
	W8	1236.09	317.64	311.38	315.52	312.75
	W9	1529.08	631.74	627.75	614.09	623.03
	W10	2900.28	1252.91	1264.38	1242.76	1250.69
	W11	5595.65	2483.71	2484.81	2509.3	2485.52
	W12	12 394.67	5013.73	5046.75	4934.95	5035.25
	W13	53 528.61	11 739.9	10 172.96	10 220.42	10 377.97
	W14	201 437.4	141 490.54	188 817.35	157 429.21	154 816.67
	W15	1 533 403.47	1 508 059.15	1 544 971.51	1 527 422.26	1 527 359.93
1	**Feed Size = 5**					
	W1	127.08	3.13	3.14	3.13	3.12

(Continued on next page)

Table B-14. (continued)

		1	2	3	4	5
	W2	90.86	5.65	5.65	5.58	5.58
	W3	26.12	11.09	11.07	10.89	10.92
	W4	370.71	20.88	21	20.8	20.9
	W5	835.61	40.42	40.21	40.08	40.67
	W6	245.09	78.21	78.55	79.97	78.7
	W7	443.88	156.23	157.47	156.2	156.46
	W8	1154.83	310.35	308.62	311.91	308.9
	W9	1402.44	617.23	614.15	618.55	617.05
	W10	2769.03	1232.68	1242.59	1232.09	1240.65
	W11	5389.49	2492.06	2516.53	2513.12	2499.31
	W12	12 228.1	5099.98	5143.73	5124.04	5100.76
	W13	55 919.46	11 577.68	10 133.54	10 169.61	10 256.72
	W14	196 510.51	130 006.86	163 649.38	148 844.43	107 346.73
	W15	1 478 308.48	1 503 564.57	1 490 593.58	1 487 541.84	1 533 537.19
2	**Feed Size = 10**					
	W1	138.05	5.1	5.17	5.09	5.09
	W2	1194.15	8.93	9.03	8.99	8.9
	W3	329.83	16.49	16.71	16.7	16.47
	W4	845.71	29.22	29.38	29.43	29.05
	W5	897.61	54.42	53.29	54.74	54.04
	W6	922.91	99.22	98.55	99.06	98.29
	W7	2975.19	183.56	183.73	182.9	183.2
	W8	3330.95	350.41	354.7	348.97	354
	W9	5535.71	678.75	676.52	686.19	677.62
	W10	12 093.23	1331.23	1322.53	1304.89	1330.63
	W11	15 701.47	2732.23	2632.81	2610.13	2566.53
	W12	25 510.35	5207.1	5169.39	5157.89	5167.89
	W13	37 315.74	15 069.83	10 313.14	10 439.43	10 495.14
	W14	70 906.46	66 104.22	67 309.46	66 774.96	68 797.17
	W15	147 629.33	148 021.99	148 232.12	146 836.54	150 915.87
2	**Feed Size = 20**					
	W1	138.03	5.14	5.15	5.1	5.12
	W2	1170.02	9.12	8.99	9.1	9.06
	W3	193.54	16.93	16.8	16.81	17.27
	W4	815.18	29.83	30.04	29.63	29.75
	W5	751.61	55.15	53.86	54.48	55.04
	W6	1195.29	97.88	96.89	98.78	98.19
	W7	3061.58	184.07	185.8	183.24	183.37
	W8	3244.91	350.23	349.85	350.29	353.35
	W9	5677.03	676.34	671.67	673.3	683.49
	W10	11 663.83	1310.18	1318.83	1310.02	1316.41
	W11	15 558.91	2693.88	2595.82	2624.4	2659.85
	W12	25 323.41	5281.32	5226.35	5221.37	5131.74
	W13	36 676.55	14 745.34	10 525.26	10 534.32	10 635.81
	W14	70 853.44	66 926.09	67 810.83	67 738.64	67 890.5
	W15	149 104.71	147 361	148 037.87	147 883.4	145 550.32
2	**Feed Size = 5**					
	W1	138.1	5.12	5.13	5.17	5.1
	W2	1170.77	8.96	8.9	8.76	8.9
	W3	168.03	16.73	16.57	16.82	16.68
	W4	319.12	29.54	30.18	29.59	29.08
	W5	918.27	54.1	53.6	53.89	54.54

(Continued on next page)

Table B-14. (continued)

		1	2	3	4	5
	W6	1500.26	97.05	96.62	97.83	98.04
	W7	3028.7	184.84	182.7	183.83	183.34
	W8	3271.64	351.64	355.21	351.43	349.41
	W9	5638.15	670.5	671.71	661.58	665.28
	W10	12 976.4	1302.61	1316.55	1319.87	1321.11
	W11	15 654.11	2668.18	2600.06	2589.62	2576.26
	W12	26 935.61	5586.14	5234.13	5202.08	5201.74
	W13	38 354.42	14 570.34	10 457.65	10 378.5	10 522.55
	W14	70 406.69	67 639.53	67 611.87	69 189.22	67 981.44
	W15	157 774.88	149 921.91	146 654.66	149 380.24	147 233.26
3	**Feed Size = 10**					
	W1	227.16	8	7.95	7.95	7.87
	W2	1199.85	14.33	14.3	14.26	14.33
	W3	969.67	26.89	26.97	26.83	27.05
	W4	1937.23	48.31	47.15	47.59	47.69
	W5	2333.57	87.22	86.06	86.53	86.46
	W6	6061.26	156.83	154.71	156.21	155.49
	W7	8838.55	289.83	280.72	276.73	279.11
	W8	15 344.79	507.16	506.7	506.3	502.32
	W9	20 603.1	1030.97	918.62	911.61	921.55
	W10	16 659.76	2098.95	1687.14	1687.72	1697.41
	W11	54 583.02	3226.84	3121.88	3143.56	3142.4
	W12	113 608.72	14 705.97	5995.83	5993.02	6022.88
	W13	142 830.32	47 111.96	11 877.22	11 874.57	11 700.95
	W14	195 435.08	188 220.19	189 170.24	189 949.34	189 613.38
	W15	294 391.01	290 758.45	290 940.07	291 768.1	291 597.21
3	**Feed Size = 20**					
	W1	227.17	7.9	7.94	7.97	7.91
	W2	1188.05	14.3	14.19	14.23	14.11
	W3	993.57	26.96	26.78	26.84	26.77
	W4	1715.01	47.8	47.34	47.28	47.64
	W5	2502.1	87.43	86.85	87.2	86.98
	W6	5818.2	195.92	156.43	157.45	156.03
	W7	8442.34	282.05	280.58	283.5	282.76
	W8	15 533.77	581.09	508.51	505.6	509.74
	W9	20 436.11	996.54	921.23	921.36	914.01
	W10	17 864.3	1957.92	1711.33	1690.21	1700.61
	W11	54 826.49	3801.89	3203.04	3195.13	3192.61
	W12	112 539.22	13 122.7	6114.55	6094.24	6127.1
	W13	143 790.28	48 527.87	12 175.79	12 294.47	12 265.56
	W14	195 675.02	188 411.75	189 488.64	189 887.72	190 393.39
	W15	295 458.87	293 411.15	292 556.64	292 097.53	292 306.8
3	**Feed Size = 5**					
	W1	225.99	8.01	7.87	7.87	7.91
	W2	1213.55	14.28	13.97	14.1	13.96
	W3	970.96	26.78	26.51	26.55	26.54
	W4	2076.98	47.4	47.06	47.21	47.05
	W5	2347.7	86.68	85.2	87.21	86.36
	W6	6590.58	153.31	151.45	155	151.56
	W7	8986.77	303.52	277.19	278.18	277.34
	W8	15 119.9	506.71	498.84	495.06	494.29
	W9	22 839.36	1103.45	894.48	907.59	906.48

(Continued on next page)

Table B-14. (continued)

		1	2	3	4	5
	W10	19 261.25	1717.69	1663.01	1644.83	1653.73
	W11	69 834.67	3329.12	3062.89	3119.48	3116.56
	W12	130 355.51	13 818.9	5975.06	6051.15	5986.81
	W13	157 848.42	49 905.28	11 637.86	11 656.5	11 770.42
	W14	218 520.25	192 188.49	191 027.72	189 745.42	190 483.53
	W15	328 668.65	298 260.4	292 675.15	291 934.93	291 187.44

Bibliography

About Gutenberg (2011). Project Gutenberg. URL: http://www.gutenberg.org/wiki/Gutenberg:About (visited on Mar. 31, 2013).

Adam, Nabil R., Bharat K. Bhargava, and Yelena Yesha, eds. (1995). *Digital Libraries Current Issues*. Vol. 916. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag. DOI: 10.1007/BFb0026845.

Apache HTTP Server Version 2.2 (2012). *ab - Apache HTTP server benchmarking tool*. The Apache Software Foundation. URL: http://httpd.apache.org/docs/2.2/programs/ab.html (visited on Feb. 14, 2012).

Apache Solr (2012). The Apache Software Foundation. URL: http://lucene.apache.org/solr/ (visited on Dec. 23, 2012).

Arms, William Y. (July 1995). "Key Concepts in the Architecture of the Digital Library". In: *D-Lib Magazine. The Magazine of Digital Library Research* 1.1. DOI: 10.1045/july95-arms.

Arms, William Y. (2001). "Libraries, Technology, and People". In: *Digital Libraries*. Ed. by William Y. Arms. 2nd. Cambridge, Massachusetts: The MIT Press. Chap. Chapter 1, pp. 1–20.

Arms, William Y., Christophe Blanchi, and Edward A. Overly (Feb. 1997). "An Architecture for Information in Digital Libraries". In: *D-Lib Magazine. The Magazine of Digital Library Research* 3.2. DOI: 10.1045/february97-arms.

Bainbridge, David et al. (2004). "Dynamic Digital Library Construction and Configuration". In: *Research and Advanced Technology for Digital Libraries*. Ed. by Rachel Heery and Liz Lyon. Springer Berlin / Heidelberg, pp. 1–13. DOI: 10.1007/978-3-540-30230-8_1.

Bainbridge, David et al. (2009). "Stress-Testing General Purpose Digital Library Software". In: *Research and Advanced Technology for Digital Libraries*. Ed. by Maristella Agosti et al. Springer Berlin Heidelberg, pp. 203–214. DOI: 10.1007/978-3-642-04346-8_21.

Baldonado, Michelle et al. (1997). "The Stanford Digital Library metadata architecture". In: *International Journal on Digital Libraries* 1.2, pp. 108–121. DOI: 10.1007/s007990050008.

Beagrie, Neil et al. (2002). *Trusted Digital Repositories: Attributes and Responsibilities. An RLG-OCLC Report*. Mountain View, CA: Research Libraries Group. URL: http://www.oclc.org/

resources/research/activities/trustedrep/repositories.pdf (visited on Sept. 13, 2012).

Berners-Lee, Tim, Roy Fielding, and Larry Masinter (2005). *RFC 3986. Uniform Resource Identifier (URI): Generic Syntax*. The Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3986.txt (visited on Mar. 31, 2013).

Borgman, Christine L, Ingeborg Solvberg, and László Kovács (2002). "Fourth DELOS workshop. Evaluation of digital libraries: Testbeds, measurements, and metrics". In: *Budapest: Hungarian Academy of Sciences*.

Borthakur, Dhruba (2007). *The Hadoop Distributed File System: Architecture and Design*. URL: http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf (visited on Oct. 25, 2012).

Boyko, Andy et al. (2012). *The BagIt File Packaging Format (V0.97)*. Version 0.97. Internet Engineering Task Force. URL: http://tools.ietf.org/html/draft-kunze-bagit (visited on Oct. 11, 2012).

Bray, Tim et al. (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation 26 November 2008*. URL: http://www.w3.org/TR/xml (visited on Nov. 19, 2012).

Candela, Leonardo et al. (Mar. 2007). "Setting the Foundations of Digital Libraries. The DELOS Manifesto". In: *D-Lib Magazine. The Magazine of Digital Library Research* 13.3/4. DOI: 10.1045/march2007-castelli.

Candela, Leonardo et al. (2008). *The DELOS Digital Library Reference Model. Foundations for Digital Libraries*. Version 0.98. DELOS Network of Excellence on Digital Libraries. URL: http://eprints.port.ac.uk/4104 (visited on Jan. 9, 2012).

Chacon, Scott (2009). *Pro Git*. Ed. by Tiffany Taylor. 1st. New York: Apress. URL: http://git-scm.com/book (visited on Oct. 11, 2012).

Conklin, Jeffrey E. and Burgess K.C. Yakemovic (Sept. 1991). "A Process-Oriented Approach to Design Rationale". In: *Human-Computer Interaction* 6.3, pp. 357–391. DOI: 10.1207/s15327051hci0603&4_6.

Crawford, Kaitlyn, Marco Lawrence, and Joanne Marston (2012). *School of Rock Art*. University of Cape Town. URL: http://pubs.cs.uct.ac.za/honsproj/cgi-bin/view/2012/crawford_lawrence_marston.zip/index.html (visited on Feb. 14, 2012).

D-Lib Working Group on Digital Library Metrics (1998). URL: http://www.dlib.org/metrics/public/index.html (visited on Mar. 30, 2013).

DELOS Workshop on the Evaluation of Digital Libraries (2004). URL: http://dlib.ionio.gr/wp7/workshop2004.html (visited on Mar. 30, 2013).

Dempsey, Lorcan and Stuart L. Weibel (July 1996). “The Warwick Metadata Workshop: A Framework for the Deployment of Resource Description”. In: *D-Lib Magazine. The Magazine of Digital Library Research* 2.7/8. DOI: 10.1045/july96-weibel.

Don, Katherine (2006). *Greenstone3: A modular digital library*. URL: http://www.greenstone.org/docs/greenstone3/manual.pdf (visited on Apr. 1, 2013).

Draft Standard for Learning Object Metadata (2002). *Final Draft Standard*. Institute of Electrical and Electronics Engineers. URL: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf (visited on Sept. 13, 2012).

DSpace Wiki (2013). *DSpace 3.x Documentation*. DuraSpace. URL: https://wiki.duraspace.org/display/DSDOC3x/DSpace+3.x+Documentation (visited on Mar. 30, 2013).

Dublin Core Metadata Element Set, Version 1.1 (1999). Dublin Core Metadata Initiative. URL: http://www.dublincore.org/documents/dces (visited on Jan. 9, 2012).

Elmasri, Ramez and Shamkant B. Navathe (2008). “The Relational Data Model and Relational Database Constraints”. In: *Fundamentals Of Database Systems, 5/E*. Ed. by Matt Goldstein. 6th. Pearson Education Inc. Chap. Chapter 3.

ETD-db: Home (2012). URL: http://scholar.lib.vt.edu/ETD-db/index.shtml (visited on Oct. 29, 2012).

Fedora Performance and Scalability Wiki (2012). URL: http://fedora.fiz-karlsruhe.de/docs (visited on Mar. 29, 2013).

Fox, Edward A. et al. (Apr. 1995). “Digital Libraries”. In: *Communications of the ACM* 38.4, pp. 22–28. DOI: 10.1145/205323.205325.

Frew, James et al. (1998). “The Alexandria Digital Library Architecture”. In: *Research and Advanced Technology for Digital Libraries*. Springer Berlin Heidelberg, pp. 61–73. DOI: 10.1007/3-540-49653-X_5.

Fuhr, Norbert et al. (2001). “Digital Libraries: A Generic Classification and Evaluation Scheme”. In: *Research and Advanced Technology for Digital Libraries*. Ed. by Panos Constantopoulos and Ingeborg T. Slvberg. Springer Berlin Heidelberg, pp. 187–199. DOI: 10.1007/3-540-44796-2_17.

Fuhr, Norbert et al. (2007). “Evaluation of digital libraries”. In: *International Journal on Digital Libraries* 8.1, pp. 21–38. DOI: 10.1007/s00799-007-0011-z.

Fulmer, Jeff (2012). *Siege Home*. URL: http://www.joedog.org/siege-home/ (visited on Dec. 23, 2012).

Gantz, John F. et al. (2008). *The Diverse and Exploding Digital Universe. An Updated Forecast of Worldwide Information Growth Through 2011*. URL: http://www.emc.com/collateral/

analyst-reports/diverse-exploding-digital-universe.pdf (visited on Oct. 27, 2012).

Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung (Oct. 2003). "The Google File System". In: *ACM SIGOPS Operating Systems Review* 37.5, pp. 29–43. DOI: 10.1145/1165389.945450.

Gilbert, Seth and Nancy Lynch (June 2002). "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services". In: *ACM SIGACT News* 33.2, pp. 51–59. DOI: 10.1145/564585.564601.

Git for Computer Scientists (2010). Tv's cobweb. URL: http://eagain.net/articles/git-for-computer-scientists/index.html (visited on Oct. 11, 2012).

Glaser, Barney (1978). *Theoretical sensitivity: Advances in the methodology of grounded theory*. Mill Valley, CA: Sociology Press.

Glaser, Barney (1992). *Basics of Grounded Theory Analysis*. Mill Valley, CA: Sociology Press.

Gohr, Andreas (2004). *DokuWiki. It's better when it's simpler*. URL: https://www.dokuwiki.org/dokuwiki (visited on Oct. 11, 2012).

Gonçalves, Marcos André, Robert K. France, and Edward A. Fox (2001). "MARIAN : Flexible Interoperability for Federated Digital Libraries". In: *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*. Ed. by Panos Constantopoulos and Ingeborg Sølvberg. London, United Kingdom. DOI: 10.1007/3-540-44796-2_16.

Gonçalves, Marcos André et al. (Apr. 2004). "Streams, Structures, Spaces, Scenarios, Societies (5s): A Formal Model for Digital Libraries". In: *ACM Transactions on Information Systems* 22.2, pp. 270–312. DOI: 10.1145/984321.984325.

Griffin, Stephen M. (July 1998). "NSF/DARPA/NASA Digital Libraries Initiative". In: *D-Lib Magazine. The Magazine of Digital Library Research* 4.7/8. DOI: 10.1045/july98-griffin.

Gutteridge, Christopher (2002). "GNU EPrints 2 Overview". In: *11th Panhellenic Academic Libraries Conference*. URL: http://eprints.soton.ac.uk/256840/ (visited on Sept. 13, 2012).

Hammer, Stuart and Miles Robinson (2011). *Bonolo*. University of Cape Town. URL: http://pubs.cs.uct.ac.za/honsproj/cgi-bin/view/2011/hammar_robinson.zip/Website/index.html (visited on Feb. 14, 2012).

Hart, Michael (1992). *Project Gutenberg. The History and Philosophy of Project Gutenberg*. URL: http://www.gutenberg.org/wiki/Gutenberg:The_History_and_Philosophy_of_Project_Gutenberg_by_Michael_Hart (visited on Jan. 9, 2012).

Hillmann, Diane (2005). *Using Dublin Core*. URL: http://dublincore.org/documents/usageguide (visited on Oct. 31, 2012).

Installing EPrints 3 via apt (Debian/Ubuntu) (2011). URL: http://wiki.eprints.org/w/Installing_EPrints_3_via_apt_(Debian/Ubuntu) (visited on Mar. 31, 2013).

Item Level Versioning (2012). *DSpace 3.x Documentation*. URL: https://wiki.duraspace.org/display/DSDOC3x/Item+Level+Versioning (visited on Mar. 31, 2013).

Janée, Greg and James Frew (2002). "The ADEPT digital library architecture". In: *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*. JCDL '02. New York, NY, USA: ACM Press, pp. 342–350. DOI: 10.1145/544220.544306.

Jetty:// (2012). Eclipse Foundation. URL: http://www.eclipse.org/jetty/ (visited on Dec. 27, 2012).

Kahn, Robert and Robert Wilensky (Mar. 2006). "A framework for distributed digital object services". In: *International Journal on Digital Libraries* 6.2, pp. 115–123. DOI: 10.1007/s00799-005-0128-x.

Körber, Nils and Hussein Suleman (2008). "Usability of Digital Repository Software: A Study of DSpace Installation and Configuration". In: *Digital Libraries: Universal and Ubiquitous Access to Information*. Ed. by George Buchanan, Masood Masoodian, and SallyJo Cunningham. Vol. 5362. Springer Berlin Heidelberg, pp. 31–40. DOI: 10.1007/978-3-540-89533-6_4.

Kruchten, Philippe (2004). "An Ontology of Architectural Design Decisions in Software-Intensive Systems". In: *2nd Groningen Workshop on Software Variability*, pp. 1–8. URL: https://courses.ece.ubc.ca/~417/public/Kruchten-2004.pdf (visited on Dec. 14, 2012).

Kruchten, Philippe et al. (2005). "Building up and Exploiting Architectural Knowledge". In: *5th Working IEEE/IFIP Conference on Software Architecture*. IEEE, pp. 291–292. DOI: 10.1109/WICSA.2005.19.

Kucsma, Jason, Kevin Reiss, and Angela Sidman (Mar. 2010). "Using Omeka to Build Digital Collections: The METRO Case Study". In: *D-Lib Magazine. The Magazine of Digital Library Research* 16.3/4. DOI: 10.1045/march2010-kucsma.

Kunze, John A. et al. (2008). *Pairtrees for Object Storage*. URL: https://confluence.ucop.edu/display/Curation/PairTree (visited on Oct. 11, 2012).

Lagoze, Carl, Clifford A. Lynch, and Ron Daniel (1996). *The Warwick Framework. A Container Architecture for Aggregating Sets of Metadata*. Cornell University. HDL: 1813/7248.

Lagoze, Carl and Herbert Van de Sompel (2001). "The Open Archives Initiative: Building a Low-Barrier Interoperability Framework". In: *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*. New York, USA: ACM Press, pp. 54–62. DOI: 10.1145/379437.379449.

Lagoze, Carl et al. (2002a). *Implementation Guidelines for the Open Archives Initiative Protocol for Metadata Harvesting. Guidelines for Repository Implementers*. URL: http://www.

openarchives.org/OAI/2.0/guidelines-repository.htm (visited on Dec. 14, 2012).

Lagoze, Carl et al. (2002b). *The Open Archives Initiative Protocol for Metadata Harvesting*. URL: http://www.openarchives.org/OAI/openarchivesprotocol.html (visited on Jan. 9, 2012).

Lagoze, Carl et al. (Apr. 2006). "Fedora: an architecture for complex objects and their relationships". In: *International Journal on Digital Libries* 6.2, pp. 124–138. DOI: 10.1007/s00799-005-0130-3.

Lee, Jintae and Kum-Yew Lai (Sept. 1991). "What's in Design Rationale?" In: *Human-Computer Interaction* 6.3, pp. 251–280. DOI: 10.1207/s15327051hci0603&4_3.

Lee, Larix and Philippe Kruchten (2007). "Capturing Software Architectural Design Decisions". In: *Canadian Conference on Electrical and Computer Engineering*. IEEE, pp. 686–689. DOI: 10.1109/CCECE.2007.176.

Leuf, Bo and Ward Cunningham (2001). *The Wiki way: quick collaboration on the Web*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Lorist, Jeroen H. H. and Kees van der Meer (2001). "Standards for Digital Libraries and Archives: Digital Longevity". In: *Proceedings of the 1st International Workshop on New Developments in Digital Libraries: In Conjunction with ICEIS 2001*. ICEIS Press, pp. 89–98.

Lynch, Clifford A. (Jan. 1991). "The Z39.50 Information Retrieval Protocol: An Overview and Status Report". In: *SIGCOMM Computer Communication Review* 21.1, pp. 58–70. DOI: 10.1145/116030.116035.

MacKenzie, Matthew et al. (2006). *Reference Model for Service Oriented Architecture 1.0. OASIS Standard, 12 October 2006*. Organization for the Advancement of Structured Information Standards. URL: http://docs.oasis-open.org/soa-rm/v1.0 (visited on Sept. 13, 2012).

MacLean, Allan et al. (Sept. 1991). "Questions, Options, and Criteria: Elements of Design Space Analysis". In: *Human-Computer Interaction* 6.3, pp. 201–250. DOI: 10.1207/s15327051hci0603&4_2.

Maly, Kurt J. et al. (2004). "Light-Weight Communal Digital Libraries". In: *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*. JCDL '04. New York, NY, USA: ACM Press, pp. 237–238. DOI: 10.1145/996350.996403.

Misra, Dharitri, James Seamans, and George R. Thoma (2008). "Testing the scalability of a DSpace-based archive". In: *Proceedings of the IS&T Archiving*, pp. 36–40.

Nicola, Matthias and Jasmi John (2003). "XML Parsing: A Threat to Database Performance". In: *Proceedings of the twelfth international conference on Information and knowledgemanagement*. CIKM'03. New York, NY, USA: ACM Press, pp. 175–178. DOI: 10.1145/956863.956898.

Nielsen, Jakob (1993). *Response Times: The 3 Important Limits*. URL: http://www.nngroup.com/articles/response-times-3-important-limits (visited on Mar. 29, 2013).

Paepcke, Andreas et al. (Apr. 1998). "Interoperability for Digital Libraries Worldwide". In: *Communications of the ACM* 41.4, pp. 33–42. DOI: 10.1145/273035.273044.

Park, Sung Hee et al. (2011). "VT ETD-db 2.0: Rewriting the ETD-db System". In: *14th International Symposium on Electronic Theses and Dissertations*. Cape Town, South Africa. URL: http://dl.cs.uct.ac.za/conferences/etd2011/papers/etd2011_park.pdf.

Paskin, Norman (2005). "Digital Object Identifiers for scientific data". In: *Data Science Journal* 4, pp. 12–20. DOI: 10.2481/dsj.4.12.

Paskin, Norman (2010). "Digital object identifier (DOI) system". In: *Encyclopedia of library and information sciences* 3, pp. 1586–1592. DOI: 10.1081/E-ELIS3-120044418.

Pepe, Alberto et al. (2005). "CERN Document Server Software: the integrated digital library". In: *9th ICCC International Conference on Electronic Publishing*. Ed. by Milena Dobreva and Jan Engelen. Leuven, Belgium, pp. 297–302. URL: http://cds.cern.ch/record/853565/.

Phiri, Lighton and Hussein Suleman (July 2012). "In Search of Simplicity: Redesigning the Digital Bleek and Lloyd". In: *DESIDOC Journal of Library & Information Technology* 32.4, pp. 306–312. URL: http://publications.drdo.gov.in/ojs/index.php/djlit/article/view/2524 (visited on Oct. 25, 2012).

Phiri, Lighton et al. (2012). "Bonolo: A General Digital Library System for File-Based Collections". In: *Proceedings of the 14th International Conference on Asia-Pacific Digital Libraries*. Ed. by Hsin-Hsi Chen and Gobinda Chowdhury. Berlin, Heidelberg: Springer Berlin / Heidelberg, pp. 49–58. DOI: 10.1007/978-3-642-34752-8_6.

Raymond, Erick Steven (2004). "Textuality. The Importance of Being Textual". In: *The Art of Unix Programming*. Ed. by Brian W. Kernighan. 1st. Boston, Massachusetts: Addison-Wesley Professional. Chap. Chapter 5, pp. 105–111. URL: http://www.faqs.org/docs/artu (visited on Oct. 17, 2012).

Saaty, Thomas L. (2008). "Decision Making with the Analytic Hierarchy Process". In: *International Journal of Services Sciences*. DOI: 10.1504/IJSSci.2008.01759.

Sadalage, Pramod J. and Martin Fowler (2012). "Why NoSQL?" In: *NoSQL Distilled. A Brief Guide to the Emerging World of Plyglot Persistence*. Ed. by Pramod J. Sadalage and Martin Fowler. 1st. Boston, Massachusetts: Addison-Wesley Professional. Chap. Chapter 1, pp. 1–12.

Sears, Russell, Catharine van Ingen, and Jim Gray (2007). "To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?" In: *CoRR* abs/cs/0701168. arXiv:cs/0701168. (Visited on Feb. 17, 2012).

Skotnes, Pippa (2007). *Claim to the Country - The Archive of Wilhelm Bleek and Lucy Lloyd*. Johannesburg, South Africa: Jacana Media.

Smith, MacKenzie et al. (Jan. 2003). “DSpace. An Open Source Dynamic Digital Repository”. In: *D-Lib Magazine* 9.1. DOI: 10.1045/january2003-smith.

Solr Wiki (2012a). *Data Import Request Handler*. URL: http://wiki.apache.org/solr/DataImportHandler (visited on Dec. 25, 2012).

Solr Wiki (2012b). *XsltUpdateRequestHandler*. URL: http://wiki.apache.org/solr/XsltUpdateRequestHandler (visited on Feb. 10, 2013).

Strand, Eric J., Rajiv P. Mehta, and Raju Jairam (Sept. 1994). “Applications Thrive on Open Systems Standards”. In: *StandardView* 2.3, pp. 148–154. DOI: 10.1145/202749.202757.

Suleman, Hussein (2002). *OAI-PMH2 XMLFile File-based Data Provider*. URL: http://www.dlib.vt.edu/projects/OAI/software/xmlfile/xmlfile.html (visited on Dec. 23, 2012).

Suleman, Hussein (2007). “Digital Libraries Without Databases: The Bleek and Lloyd Collection”. In: *Proceedings of the 11th European Conference on Research and Advanced Technology for Digital Libraries*. Ed. by László Kovács, Norbert Fuhr, and Carlo Meghini. Berlin, Heidelberg: Springer-Verlag, pp. 392–403. DOI: 10.1007/978-3-540-74851-9_33.

Suleman, Hussein (2008). “An African Perspective on Digital Preservation”. In: *Proceedings of the International Workshop on Digital Preservation of Heritage and Research Issues in Archiving and Retrieval*. Kolkata, India.

Suleman, Hussein (2010). “Interoperability in Digital Libraries”. In: *E-Publishing and Digital Libraries: Legal and Organizational Issues*. Ed. by Ioannis Iglezakis, Tatiana-Eleni Synodinou, and Sarantos Kapidakis. IGI Global. Chap. Chapter 2, pp. 31–47. DOI: 10.4018/978-1-60960-031-0.ch002.

Suleman, Hussein et al. (2010). “Hybrid Online-Offline Digital Collections”. In: *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*. Bela Bela, South Africa: ACM Press, pp. 421–425. DOI: 10.1145/1899503.1899558.

Tansley, Robert, Mick Bass, and MacKenzie Smith (2003). “DSpace as an open archival information system: Current status and future directions”. In: Lecture Notes in Computer Science 2769. Ed. by Traugott Koch and Ingeborg Sø lvberg, pp. 446–460. DOI: 10.1007/978-3-540-45175-4_41.

Tansley, Robert et al. (May 2003). “The DSpace Institutional Digital Repository System: Current Functionality”. In: *Proceedings of the 2003 Joint Conference on Digital Libraries*. Houston, United States: Institute of Electrical and Electronic Engineers, pp. 87–97. DOI: 10.1109/JCDL.2003.1204846.

The Apache HTTP Server Project (2012). The Apache Software Foundation. URL: http://httpd.apache.org (visited on Mar. 30, 2013).

The Digital Bleek and Lloyd (2007). URL: http://lloydbleekcollection.cs.uct.ac.za/ (visited on Oct. 11, 2012).

Ubuntu 12.04.2 LTS (Precise Pangolin) (2012). URL: http://releases.ubuntu.com/precise/ (visited on Apr. 1, 2013).

Vesely, Martin et al. (2004). "CERN Document Server: Document Management System for Grey Literature in a Networked Environment". In: *Publishing Research Quarterly* 20 (1), pp. 77–83. DOI: 10.1007/BF02910863.

What is Wiki (1995). URL: http://wiki.org/wiki.cgi?WhatIsWiki (visited on Mar. 30, 2013).

Wiltshire, Nicolas (2011). *Spatial analysis of archaeological sites in the Western Cape using an integrated digital archive*. University of Cape Town. URL: http://uctscholar.uct.ac.za/R/N8XKKNNCY76DM8GQG33X3C8LCDJ7N6MUKFFKAHMT67HVDJN9TT-03715?func=results-brief (visited on Feb. 14, 2012).

Winer, Dave (2007). *RSS Advisory Board. RSS 2.0 Specification*. URL: http://www.rssboard.org/rss-specification (visited on Sept. 10, 2012).

Witten, Ian H., David Bainbridge, and Stefan J. Boddie (2001). "Greenstone: open-source digital library software with end-user collection building". In: *Online Information Review* 25.5, pp. 288–298. DOI: 10.1108/14684520110410490.

Index