Honours Project Report

Bonolo: The Curator Interface

Miles Robinson
mrobinson@cs.uct.ac.za

Supervised by: Dr Hussein Suleman
hussein@cs.uct.ac.za

	Category	Min	Max	Chosen
1	Requirement Analysis and Design	0	20	15
2	Theoretical Analysis	0	25	0
3	Experiment Design and Execution	0	20	10
4	System Development and Implementation	0	15	15
5	Results, Findings and Conclusion	10	20	10
6	Aim Formulation and Background Work	10	15	10
7	Quality of Report Writing and Presentation	10		10
8	Adherence to Project Proposal and Quality of Deliverables	10		10
9	Overall General Project Evaluation	0	10	0
Total Marks		**80**		**80**

Department of Computer Science
University of Cape Town

2011

Abstract

The introduction of digital repository systems has provided a framework to aid in the preservation and dissemination of digital objects. Digital repository systems are traditionally based around databases or other customised data storage structures. While these storage structures do possess advantages in terms of performance, they have limitations in the preservation and sharing of data. This is due to the data kept in the storage structures requiring specific tools so to become human readable. One solution to this problem is to build a digital repository system using a simple data storage structure such as a file hierarchy. There are, however, concerns with the usability and performance of digital repository systems that do not make use of databases. This report describes how a digital repository management system supporting a high level of usability was built around a simple file hierarchy, with an emphasis on effective and efficient curation of digital objects. Initial results have shown that the usability and performance of this system was indeed not compromised when using structured file hierarchies.

Keywords

Digital libraries, digital repository systems, file-based digital repository, digital curation, preservation.

Acknowledgements

Firstly I would like to thank my supervisor, Dr Hussein Suleman, for his involvement in and commitment to this project. You were always available when needed and provided constructive feedback at every step of the way, even if it was sent at 3am. Your approach as a supervisor was exactly what I needed to keep motivated, and I am extremely grateful for everything you have put in over the past 6 months.

I would like to thank Lighton Phiri, a Masters student, who provided us with samples of the Bleek and Lloyd collection to test with. Your input to this project was highly appreciated, and I hope the results found in this report are of benefit to your Masters research.

To my project partner and friend– Stuart Hammar – I cannot think of anyone I would rather have worked with than you on this project. You were always an email or a g-chat away on those late stressful nights, and we shared more ridiculous moments than I care to remember.

To the Honours class of 2011 – I would like to thank all of you for the year we shared together. I am confident each one of you has a bright future, and I look forward to meeting up with you in the future

To all the people who helped us from focus groups, demonstrations and evaluations – this project could not have been completed without you. Seriously!

My digsmates and other friends – without you guys, the last four years of my life at varsity would have been really bland. Although we may be going our separate ways, I look forwards to long friendships with all of you.

Last, but definitely not least, I would like to thank my family. To my parents, Anthony and Tine, for your unwavering belief in me and the sacrifices you have made to get me through University, please know that it is never taken for granted! Finally to my brother Wesley, you genuinely do inspire me, and I would like to thank you for that, as well as helping me out with editing and finding people for evaluations. You are the best, gg no re!

Table of Contents

List of Figures

Chapter 1

Introduction

Although history is concerned with past events, it provides context and identity to current and future generations [Stearns, 1998]. Furthermore, history provides the necessary means to view current societies, trends and patterns in comparison to those of the past. Therefore, the importance of preserving history cannot be overstated. Part of this history is the culture and heritage associated with communities. These are essential tools for studying people and their roles in the societies around them.

The preservation of cultural and heritage data is particularly important on the African continent. Africa possesses a rich and diverse cultural history as it was home to the oldest communities and societies that have been discovered on Earth [Herlin, 2003]. However, many of the ancient cultures and customs of the communities around Africa and the rest of the world have been lost over the centuries due to extinction and colonisation. Even today the customs, knowledge and culture of communities are under the threat of being lost due to increasing technological and Western influences.

One solution to this problem is to document and translate the beliefs, knowledge and cultures of communities whose previous way of life is under threat. These documents can then be digitised for distribution and preservation. The Lloyd and Bleek collection is one such example of where this has occurred. As a result of the efforts of Lucy Lloyd and Wilhelm Bleek, a collection has been made that allows people of current and future generations to get a glimpse into a part of the |xam and !kun Bushman way of life [Skotness, 2011].

The introduction of digital repository systems provides further incentive for these resources to be digitalised, as they provide a means of aiding in the preservation of the culture as in the case of the Lloyd and Bleek collection. In addition to the advantage of preservation, digital repository systems also provide a framework to allow a much larger number of people to experience and learn from other cultures.

1.1 The Bonolo project

The Bonolo[1] project seeks to build an online digital repository system on top of a simple file hierarchy as its data store. This differs from the majority of existing digital repository systems, which use

[1] Sesotho for “simple”

databases or other customised data storage structures. While these storage structures do possess advantages in terms of performance, they have limitations in the preservation and sharing of data.

The digital repository system that has been built allows users to access and interact with the contents of the digital repository. It also allows curators to manage the collections within the digital repository. This system provides users and curators with these capabilities of traditional digital repository systems for any simple file storage layout that follows the simplyCT format. More details on the simplyCT format are provided in *section 1.2.5 Data Description*.

Furthermore, the digital repository system that is to be built should be lightweight, easy to use and easy to install. By meeting these objectives, the system will make it easier for more to be built.

1.1.1 Motivation

Many digital collections worldwide are stored in digital repository systems that are based around databases or other complex storage systems. These repository systems aid in the preservation of these digital objects through storing copies. However, because of storage systems being complex, viewing these objects is often impossible without the tools provided by the particular digital repository system.

This approach has two disadvantages. The first is that it does not allow different digital repository systems to share content as each system is likely to use a different storage system. Thus, complex systems often have to be installed so that the data stored within the digital repositories can be read and used. The second is the rapid rate of technological progress resulting in software that is continuously being updated, with older versions quickly becoming out-dated. As a result there is a risk of the tools that can access these storage systems becoming obsolete and thus access to the stored data is lost.

These disadvantages highlight the need for a digital repository system that is built on a simple storage structure that can be read without the aid of tools provided by the digital repository system. The new system should also be very lightweight, requiring very little installation in order to set up a digital repository. The objective of the Bonolo project is to meet these needs.

1.1.2 The Framework

There are three separate components that form the Bonolo framework. These three components are: the Curator Interface; the User Interface; and the Repository. Figure 1.1 shows a high level view of the framework used in Bonolo. Each of these components will now be explained in more detail.

Figure 1.1: Overview of the Bonolo Framework

The Repository

The Repository is the central component of the Bonolo framework. It is where all the data that is available to both users and curators is stored, and may consist of multiple collections. The Repository is a common point accessed by both the curator interface and the user interface. Both the curator interface and the user interface have been designed in such a way that the content of the Repository is irrelevant as long as it consists of a file hierarchy that is positioned correctly in the above framework. For the purposes of this project a sample of the Lloyd and Bleek collection was provided by Lighton Phiri, a Masters Student at the University of Cape Town. This collection consists of a collection of .jpg images and metadata files that are organised into a file hierarchy that follows the simplyCT framework. The simplyCT[1] framework is a set of rules that define how files and data are to be structured in a collection so that they can be effectively used.

The User Interface

The User Interface provides a means for people to view and interact with the digital objects stored within the collection. This component of Bonolo was implemented by Stuart Hammar. The user interface provides the following functionality:

- A register and login system that makes use of live email validation.
- Search and Browse the stories in the collection using a faceted search with a filter to narrow down results.
- View the metadata associated with the stories.
- Recommend similar stories.
- Browse the books in the collection.
- Comment on items in the collection.
- Download images or entire books from the collection.
- Allow the administrator of the system to customise a configuration file that determines how the facets on the home page and on the search page are displayed.

The aim of this component is to provide users with an effective means of finding, viewing and interacting with the digital objects stored within the collection.

The Curator Interface

The Curator Interface is a Web application that attempts to allow the curators in charge to manage the digital collections remotely. This is done by providing the users of the Web application with tools to browse through the collection as well as add, delete and edit files. Furthermore, the Curator Interface will be responsible for ensuring the long term preservation of the collection stored within the digital repository. This report will focus on the Curator Interface component of the Bonolo framework.

[1] http://dl.cs.uct.ac.za/projects/simplyCT

1.2 The Curator Interface

1.2.1 Problem Statement

A key component of all digital repository systems is a means for administrators to manage the collections. In the case of most digital libraries this is done through queries and updates to a database or complex storage system. Given the objectives of the Bonolo project, a means of managing a collection based on a simple file hierarchy as a data store needs to be constructed. Additionally, the system that is to be developed should be lightweight, requiring little effort to set up a new collection.

1.2.2 Motivation

The motivation for the Curator Interface is to address the issues raised in the problem statement by building a light-weight Web application that allows curators to manage a collection of files stored in a simple file hierarchy on the server. The Curator Interface is an essential part of the Bonolo digital repository system, which has the potential to be released as an alternative to existing digital repository systems that are based on complex storage structures.

In addition, the results and findings of this investigation may lead to a new approach to the way in which digital repository systems are created and used. The simple nature of the file storage structure, as well as the lightweight nature of the system could lead to new collections being established.

1.2.3 Research Questions

The Curator Interface seeks to address the following research questions as a component of the Bonolo framework:

1. Will using a simple file hierarchy as a file store affect the overall usability of the Curator Interface?
2. What is the impact of having a hierarchical file-based data store on the performance of the Curator Interface?
3. Is the Bonolo Curator Interface comparable to those of other digital repository systems?

To answer these research questions, user evaluation as well as performance evaluation will be undertaken. The results of the user evaluation will be used to determine what effect the file store will have on the usability of the Curator Interface, as well as the comparability to other digital repository systems. The performance of the Curator Interface will be determined through testing the response times and amount of data transferred using different sized and structured collections.

1.2.4 The Framework

The goal of the Curator Interface is to provide an online tool to curators with which they can manage a collection of digital objects stored in a simple file structure. This tool therefore needs to provide curators with a set of core functions allowing them to view, add, delete and edit files in the collection. Figure 1.2 shows an overview of the Curator Interface structure, and how the various components of the framework link together.

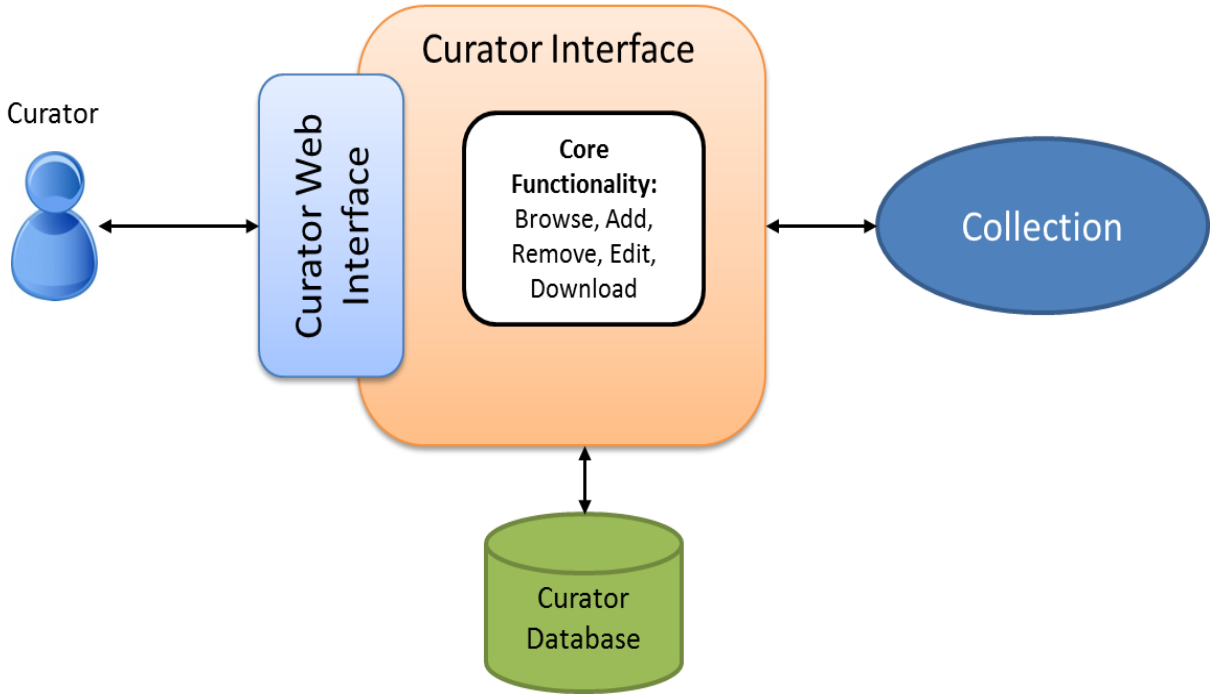

Figure 1.2: The Curator Interface Framework

Web Interface

- Provides curators with access to the digital objects stored in the collection.
- Provides curators with a means of using the tools necessary to interact with and manage the collection.

Curator Interface

- Defines the core functionality necessary for curators to manage the collection. This core functionality is as follows: Browse; Add; Remove; Edit; and Download.
- Defines the auxiliary functions that the curators have access to. The auxiliary functionality of the Curator Interface is: displaying curator account details; recent activity performed on the collection; Login; Logout; and Register.
- Makes use of the Curator Database to determine what access rights a particular curator has.
- Performs the specified actions of the curator on the collection.

Curator Database

- Contains information regarding all curator account details (i.e. username, password and e-mail).
- Contains details about which collections each curator can access.
- Contains details about which collections each user is waiting to gain curatorship over.

Collection

- All the digital objects the curator has access to. This is a subset of the Repository shown in the Bonolo framework.

1.2.5 Data Description

There are two considerations that have been made in terms of the data that is to be used in the Curator Interface. The first is the file structure of the collection in the digital repository; while the second is the data type of the actual digital objects that make up the collection. Each of these will now be discussed in more detail.

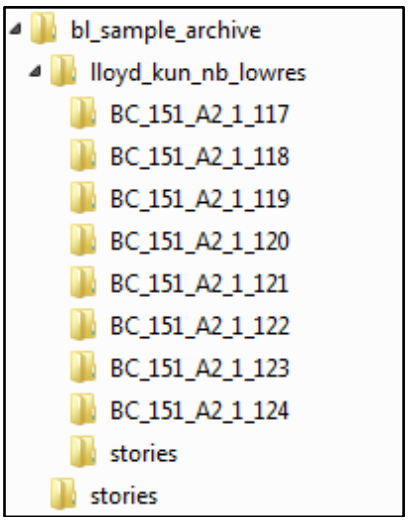

Figure 1.3: Folder Structure of the Lloyd and Bleek collection

As previously mentioned, the primary consideration of Bonolo is that it is to be built around a file hierarchy. Ideally, all aspects of Bonolo should be able to effectively work on all possible file structures. The Curator Interface is currently able to parse any file structure, and allow authorised users to browse the contents of all the files or folders. However, as explained in Chapter 3, the current system provides the best functionality when used with the simplyCT framework. For the purposes of this project, a sample of the Lloyd and Bleek Collection was provided. The structure of this collection follows the guidelines laid out by the simplyCT framework, and can be seen in Figure 1.3.

At this point, the only digital objects that are supported by Bonolo are .jpg images (Figure 1.4 shows an example image from the Lloyd and Bleek collection) and their associated metadata files. While this is a fairly large limitation of the system, the framework that has been constructed is easily extensible to allow for more data types. The metadata files associated with the digital objects also have the following restrictions: they must end with a .metadata extension; and they must be valid XML documents. If these restrictions are not met, the system will either not register them as metadata files, or not parse them correctly. Once again, the Curator Interface framework is easily extensible to allow for different metadata file extensions. However, the framework is heavily dependent on valid XML documents being used for metadata.

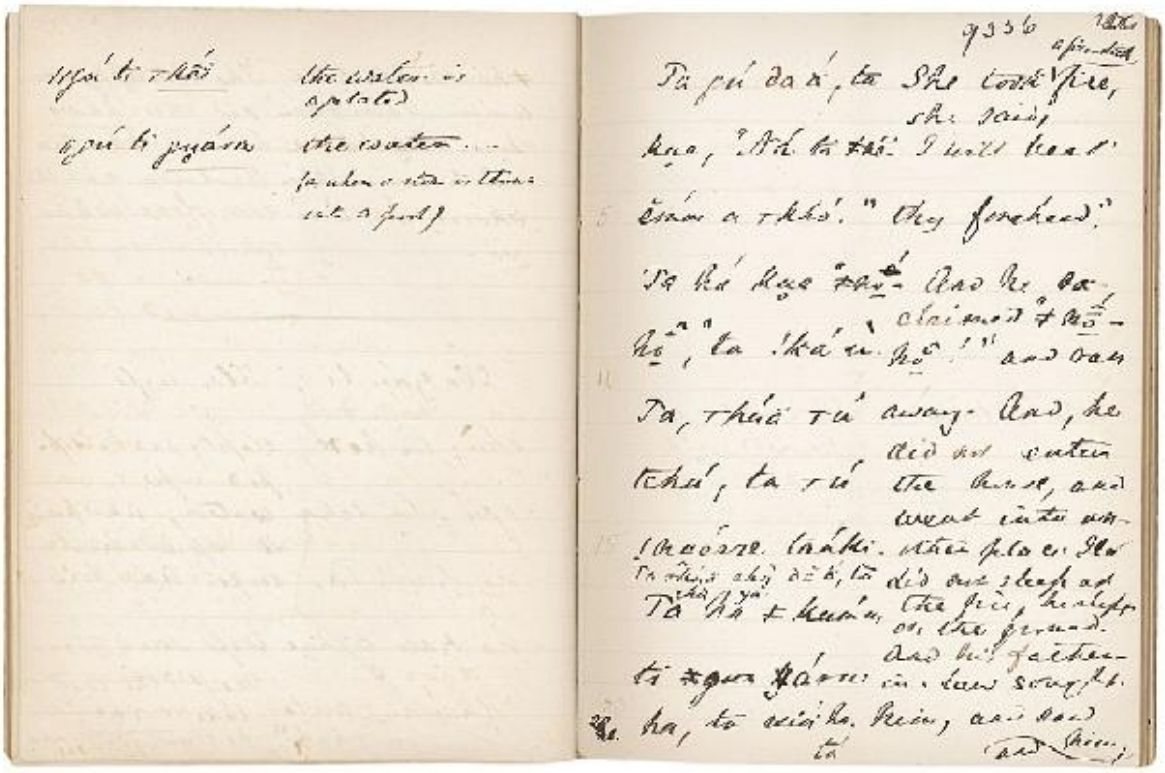

Figure 1.4: Sample image from the Lloyd and Bleek collection

1.2.6 Ethical, Professional and Legal Issues

There may be legal and ethical issues with the content stored in the collections. It is therefore important that the users of Bonolo understand and adhere to the copyright agreements imposed on the content that is being used in the collection. Furthermore, any version of the Curator Interface that is released will have to be adjusted to state who owns the copyrights to the content in its collections.

Regarding the ethical issues surrounding user evaluations, the following action was taken. Ethical clearance to allow for user testing was granted by the Faculty of Science Research Ethics Committee of the University of Cape Town. Furthermore, access to University of Cape Town students for research purposes was granted by the Department of Student Affairs. Finally, each user who

participated in a user evaluation signed a consent form allowing the use of their feedback to assess the Curator Interface.

The commons fileUpload and commons IO libraries were used in the Curator Interface and are available under the Apache Commons license [Apache Commons, 2011]. SQLite – the database tool used for managing curator access to collections – falls within the public domain and is available for use in any manner [SQLite, 2011]. Finally, jQuery falls under either the MIT License or the GNU General Public License Version 2, depending on the needs of the project [The jQuery Project, 2011].

Unless explicitly stated otherwise, the Bonolo Curator Interface is licensed under the GNU General Public License (GPL) Version 3 [GNU, 2011].

1.3 Report Outline

This report gives a description of the Bonolo Curator Interface and begins with an investigation into similar work that has been done. Following that, the design and implementation of the Curator Interface is discussed more thoroughly. The user and performance evaluations that were carried out on the Curator Interface will be explained and their results presented. Finally, the potential future work will be considered before the conclusions of the Curator Interface are drawn.

Chapter 2

Background

2.1 Introduction

Since the 1990s, research into and development of digital repositories has increased significantly [Borgman, 1999]. The progression of these digital repositories has led to many organisations worldwide adopting them to effectively manage large collections of digital objects. These digital repository systems have traditionally shared a close link with database systems [Suleman, 2007a]. These database systems have proven to be useful to digital repository systems as they provide the mechanisms to efficiently add, delete, update and retrieve items in a collection. However, although there are advantages provided by database systems for managing a collection, there are also certain drawbacks. Systems that make extensive use of databases are often not well suited for portability as they require installation and are not usually platform-independent. Furthermore, the data inside these database systems is usually stored in a binary format, meaning that without the use of the tools provided by the system, it is unreadable to humans. Ultimately this means that the long-term preservation of the digital objects in the database is not ensured because if the tools to read the database are lost, so are the contents of the database.

One solution that can address the problems of databases is the use of a file hierarchy as a data storage structure [Suleman, 2007a]. This method ensures that digital objects stored in the repository can be viewed without the tools provided by the digital repository system. However, little research has been done into the concept of building a digital repository system on top of simple data storage structures such as a file hierarchy.

This chapter of the report will establish the background to the project by looking into the advantages offered by traditional digital repository systems. Following that, a more specific discussion into the background of this project will be conducted. This will be done by investigating the management systems of some well-established digital repositories to provide an understanding of the objectives a curator interface should meet. Lastly, some digital repository systems built around simple file hierarchies as repositories will be explored.

2.2 Advantages of Digital Repository Systems

The increase in use and popularity of digital repository systems is primarily due to the numerous advantages they offer to those in charge of digital collections. At their most basic level, digital repository systems provide a means of storage and access to various digital objects. While these

functions are useful, the real value digital repository systems offer comes from the functionality they support beyond storage and access [Geisler et al., 2002]. It was pointed out by Cleveland [1998] that building effective digital libraries would involve overcoming the issues surrounding: technical architecture; building digital collections; digitisation; metadata; naming, identifiers and persistence; copyright; and preservation. Since then, many of these have been addressed to some degree in the digital repository systems that are available for use today. As a result of addressing these issues, a framework to promote organisation, preservation and accessibility is provided by digital repositories.

2.2.1 Organisation

The organisation of digital objects stored in a digital repository is important as it allows a variety of user services (for example, searching) and internal services (for example, preservation) to be implemented. Organisation is offered by digital repository systems and is achieved through the use of metadata, as well as the naming of digital objects.

Metadata

Metadata is additional information associated with a digital object that is used to describe the contents and attributes of that digital object [Kuny and Cleveland, 1998]. The metadata system being used in the digital repository should provide a framework that allows any stored digital object to be effectively described. The metadata associated with objects in the digital repository is typically split into descriptive and administrative metadata [Gartner, 2008]. Descriptive metadata describes the digital object, and may follow the Dublin Core schema. The Dublin Core schema is the most widely accepted schema, and therefore can be interpreted by the widest range of systems [Gartner, 2008]. Administrative metadata is further split into technical metadata, rights metadata, and preservation metadata. Each of these will typically follow a well accepted metadata schema for their respective purpose [Gartner, 2008].

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) can provide further benefits to the organisation of a digital repository [Lagoze & Van de Sompel, 2003]. OAI-PMH allows third parties to harvest the metadata and build services that add value to it [Lagoze & Van de Sompel, 2003]. Interoperability between different digital repositories is thus facilitated through the use of OAI-PMH. This allows for the interaction of existing services, and provides potential to create new services for particular infrastructures. Furthermore, contributions to a larger, general system are made possible through interoperability [Payette et al., 1999].

Naming

The name attached to a digital object serves as a unique identifier to that object. To ensure that this identifier is unique, a permanent, collective system for naming objects needs to be established. This has various implications, most notably that the name cannot be associated with a particular location [Cleveland, 1998]. This ensures that resources and their metadata are not lost if their locations change. To solve the issues surrounding persistent naming, various schemes have been introduced. Included in these are: PURLS – Persistent URLs; URN – Uniform Resource Name; and DOI – Digital Object Identifier System [Cleveland, 1998].

2.2.2 Preservation

The preservation of the data in the repository is another advantage offered by digital repository systems. Preservation is important as the data being stored may be very rare and valuable [Suleman, 2007a]. There are four important aspects that digital repository systems address with regard to preservation: Integrity; Structural Design; Storage Media; and Ongoing Maintenance.

Integrity

Integrity involves ensuring that important digital objects in the digital library are not modified accidentally or purposefully (unauthorised modification). Integrity is important with regard to preservation as it ensures that a digital object is not incorrectly modified during its lifetime [Jantz and Giarlo, 2005]. Several techniques can be implemented in order to ensure the integrity. Examples of these techniques include: digital signatures; audit trails; and checksums.

Structural Design

The structural design refers to both the metadata associated with the object as well as the format that the object is stored as. Digital repository systems provide the means for easily backing up the data in the collections. For example, LOCKSS (Lots of Copies Keep Stuff Safe), a peer-to-peer preservation system, may be integrated to enhance the preservation of the collections by keeping copies distributed over a number of locations [Santhanagopalan et al., 2006].

Technologies may go out of date in the future, thus rendering the formats associated with them obsolete. To solve this problem, data could be migrated to a different format [Jantz and Giarlo, 2005]. Digital repository systems provide the tools necessary to export the data in the collections to new formats. However, this process has not yet been perfected and is still being researched and improved.

Storage Media

The rapid development in computer hardware means that a particular storage medium is likely to go out of date within a few years [Cleveland, 1998]. This means that digital objects will have to be regularly transferred from older storage media to newer ones. While undergoing the transferral process there is the risk that data is lost as the newer technology no longer has the ability to read the information (for example CD-ROM drives cannot read floppy disks). Digital repository systems have the ability to ensure that data integrity is maintained while it is being migrated between different storage media [Jantz and Giarlo, 2005].

Ongoing Maintenance

The ongoing maintenance of a digital library is the final aspect of preservation. Should a digital library go unmaintained, there is a high likelihood of data being lost or corrupted. In order to keep digital objects preserved, it is important that regular maintenance is performed. This is particularly relevant when data is to be migrated, or the system is being upgraded [Ackerman and Fielding, 1995]. Digital repository systems provide the administrators with the means of cleaning and maintaining collections, thus aiding in their preservation.

2.2.3 Accessibility

Accessibility concerns two key issues of digital repositories, specifically: access control to resources; and the availability of resources [Cleveland, 1998]. Digital repository systems implement accessibility, and therefore provide those in charge of the collections with a greater degree of control over the collections as well as means of disseminating the content of the collections.

Access Control

Access control involves restricting access to digital objects in the repository. The nature of digital objects is such that they can easily be copied and distributed without the authority of the owners [Cleveland, 1998]. As digital repository systems often do not have ownership of copyright of resources they are storing, mechanisms to ensure the actions of a user are allowed by the copyright need to be implemented. To achieve this, middleware can be implemented on the digital repository system to manage what users have access to, and what actions they are allowed to perform.

Availability

One of the key advantages offered by digital repositories is the dissemination of the objects they contains. To ensure that the objects are distributed as effectively as possible, the content of the digital repository should be made readily and economically available to the public [Waters, 1998]. Digital repository systems create an environment that allows users to easily access the content stored in the repository.

2.3 Digital Repository Management Systems

To gain a better understanding of the functionality offered to those in charge of collections, three existing digital repository systems were analysed. These three systems – Dspace[1]; Fedora[2]; and Greenstone[3] – were chosen because they are among the most common digital repository systems in use today, and would thus give a good indication of the functionality that should be supported. Each of these three systems will now be analysed with regard to the manner in which collections are managed and the functionality that is offered to the curators.

2.3.1 DSpace

DSpace is an open source digital repository system that provides the tools necessary to store, manage and access digital objects [Smith et al., 2003]. Administrators have complete control over collections, and thus play a central role in managing them. Administrators are not, however, the only ones who have the ability to manage collections. While many of the features offered by DSpace are available to anonymous users, certain functions require user authentication through registration [Tansley et al., 2003]. Registered users, called e-people, can be granted permission by the administrator to perform

[1] http://www.dspace.org/
[2] http://fedoraproject.org/
[3] http://www.greenstone.org/

certain tasks on collections. Groups of e-people with the same permissions can be constructed by the administrator to make collections easier to manage. The different permissions and the actions they permit are as follows [Tansley et al., 2003]:

- Read
    - Being aware of an object's existence, and viewing its associated metadata.
- Write
    - Modifying an object's associated metadata.
- Add
    - Adding an object to a container.
- Remove
    - Remove an object from a container.
- Workflow
    - Participate in the approval or denial of object submission to a collection.

From this list it can be seen that, with the appropriate permissions, the following functions can be performed to manage the collections in the DSpace digital repository system:

- Edit metadata
- Add files to the collection
- Add metadata to the collection
- Create folders in the collection
- Remove objects from a collection
- Approve or deny item submissions to a collection

2.3.2 Fedora

Fedora (Flexible Extensible Digital Object Repository Architecture), like DSpace, is an open source digital repository system that provides an architecture that supports the storage, management and dissemination of digital objects in a collection [Staples et al., 2003]. The Fedora digital repository system makes use of public APIs that are exposed as Web services to provide access to the collections. One of these APIs, specifically API-M, is the management service, which defines the interface for administering collections. Access control is imposed on the API-M through IP-range restriction and HTTP Basic Authentication. This ensures that unauthorised users do not make unwanted adjustments to the collections.

The following operations are made possible through the use of API-M to administer the collections [Staples et al., 2003]:

- Creating digital objects
- Adding metadata
- Modifying digital objects
    - This is done through editing metadata

- Deleting digital objects
- Importing digital objects
- Exporting digital objects
- Maintaining digital objects
    - Ensuring that digital objects are being correctly preserved
- Batch uploading

2.3.3 Greenstone

Greenstone digital library software is another open source system that allows collections to be constructed and disseminated [Witten et al., 2000]. Management of the digital objects stored in the collections is done through the Greenstone Librarian Interface (GLI) [Witten, 2004]. The GLI is a platform independent Java application, which is usually run on the same computer that hosts the Greenstone digital library server. Here, it is used to help librarians with the construction and organisation of digital information collections.

The GLI makes distinctions among users who have access to a collection by providing four different levels of control [Witten, 2004]. Assistant librarians have the lowest degree of control and only have access to the most basic features of the GLI. They are only able to add new documents and metadata to existing collections; make new collections that follow the same structure as existing collections; and update existing collections to show changes that may have occurred. Librarians can perform all the tasks of an Assistant Librarian as well as designing new collections. Library Systems Specialists are able to customise collections in complex ways in addition to the actions available to Librarians. Finally, Expert users have access to all features of the Greenstone Librarian Interface and are able to run Perl scripts and examine their output. The level of user is usually determined by the amount of experience a particular user has.

The GLI allows librarians to perform the following functions [Witten, 2004]:

- Add documents from the computer to a collection
    - Documents can be transferred from other collections on the computer. In this case, the document's associated metadata file will be added as well.
    - Add metadata to documents
- Add metadata to objects in the collection
- Construct folders in the collection
- Delete files from the collection
- Edit metadata
- Set the public availability of the collection
- Configure the collection to determine its appearance and access facilities

2.4 Digital Repository Systems Using Simple Storage Structures

While numerous digital repository systems are currently in existence, very few use simple file structures as a means for data storage. Systems that do use simple file structures include: the Bleek and Lloyd collection[1]; CALJAX[2]; and Acumen[3], each of which will now be briefly examined.

2.4.1 The Bleek and Lloyd Collection

The digital repository system designed for the Bleek and Lloyd Collection [Skotness, 2011] sought to overcome the issues with using databases by constructing an XML-centric solution [Suleman, 2007a]. This approach was arguably portable as it is platform-independent, and many of the tools required to work with XML are embedded in modern browsers. Furthermore, this solution allowed for easier processing and ensuring the long-term preservation of the data. This was possible because XML documents are human-readable and do not require specific tools to extract and adjust their contents. It was shown by Suleman [2007a] that the XML-centric approach was not only possible, but had many advantages (particularly in portability and preservation) over traditional digital repository systems using databases.

There are, however, two issues with this solution. The first is that there are concerns around the scalability of XML-centric solutions. Suleman [2007a] was able to show that acceptable performance was achieved in the Bleek and Lloyd Collection, which consists of over 14000 digital objects. Nevertheless, the scalability of XML-centric solutions is seen as a limiting factor. The second is that this solution has been specifically designed for the Bleek and Lloyd Collection. As a result, it cannot be easily extended to be used in other collections.

2.4.2 CALJAX

Suleman [2007b] noted that, with the emergence of tools such as AJAX, browsers now allow clients to perform tasks that previously had to be performed on the server. The CALJAX project investigated the feasibility of building a fully functional digital repository system that used AJAX as its base technology [Suleman et al., 2010].

The repository used in the CALJAX project consisted of binary files and their associated metadata stored in hierarchical directories [Suleman et al., 2010]. This allows the contents of the repository to be easily copied onto a removable media device and deployed elsewhere. Along with the simple file storage system used, the system developed by the CALJAX project is lightweight. It has a minimal software footprint, requires no software installation and is platform-independent. Another advantage of CALJAX is that it was designed to be used in an offline or limited bandwidth environment. The combination of these factors results in a system that is portable and can easily be deployed in a wide range of situations.

[1] http://lloydbleekcollection.cs.uct.ac.za/
[2] http://shenzi.cs.uct.ac.za/~honsproj/cgi-bin/view/2009/bowes_hirst_subrun.zip/marcbowes-honours-project-site-7cf4396/
[3] http://sourceforge.net/projects/acumendls/

2.4.3 Acumen

Another example of a lightweight digital repository system is Acumen [Loewald & DeRidder, 2010]. Acumen was designed to be a user-friendly Web application that allows anyone to create and maintain collections that end-users have access to. Like CALJAX the digital objects in the collection are stored in a file structure. However, the metadata associated with those objects is not stored alongside them as is the case with CALJAX. Instead, metadata about each object is entered into a spreadsheet which is then exported as valid XML documents. These XML documents are then stored on a Web server that Acumen can reference and access.

Management of the collections is done through the Archivist Utility in Acumen [Loewald & DeRidder, 2010].This Archivist Utility is a desktop application that is installed by the administrator of a collection. Even though Acumen is operating system independent, it does require installation and is therefore not as portable as the previous two systems. However, Acumen is significantly more flexible than both the Bleek and Lloyd collection and CALJAX.

2.5 Discussion

This chapter of the report provides a background to the project by analysing some of the research that has been done into digital repository systems. First, the advantages provided by digital repository systems were investigated. Although the primary function of these systems is to provide a means of storage and access to digital objects, it was found that the real advantages provided by digital repository systems comes from the additional functionality they support. This additional functionality is the capacity to organise, preserve and promote the accessibility of collections within the repository.

Next the management systems of DSpace, Fedora and Greenstone were investigated to determine what essential functions the Curator Interface should support. While it was found that each provided slightly differing functionality, the following core functions were supported by all of them: adding digital objects to the collection; removing digital objects from the collection; adding metadata to a particular object; and editing metadata. These functions are therefore seen as the primary functions a digital repository management system should support.

Finally, digital repository systems using a simple file structure as a storage structure were investigated. It was shown that the Bleek and Lloyd Collection, CALJAX and Acumen were all able to implement a digital repository system that was based around a simple file hierarchy as a storage structure. An XML-centric approach was taken in the Bleek and Lloyd Collection, which provides some significant advantages in terms of portability and preservation over traditional database systems. The solution used in the Bleek and Lloyd collection is, however, specific and cannot be easily extended to facilitate different collection structures. CALJAX made extensive use of modern browser technologies such as AJAX to build a highly portable system that used a file hierarchy as a storage structure. Finally, Acumen also makes use of a file hierarchy to store objects in the collection, however the associated metadata is stored as XML documents on a Web server. Although Acumen is not as portable as Bleek and Lloyd and CALJAX, it is much more flexible, providing a solution that is able to manage a wide range of different collections.

Chapter 3

Design and Implementation

3.1 Introduction

The main objective of implementation for the Curator Interface was to construct an online framework that allowed curators to effectively manage a data store kept on a server. The data store would be made up of a file hierarchy, with a very limited degree of prescribed structure. The reasoning behind having a file store without a rigid structure was to build a system that was as general as possible and could easily be adopted in a wide range of collections.

This chapter addresses the design and implementation of the Curator Interface in as much detail as possible. This is done to aid those who may wish to reconstruct a component of the Curator Interface in the future. The description of design and implementation begins by detailing the tools and technologies that were used for implementation, as well as the scope and limitations within which the project was developed. Following that, the different aspects of the design and their respective implementation will be discussed before the iterative development of the system is explained. Finally, the issues and findings that emerged from the design and implementation process will be analysed and discussed.

3.2 Technology and Tools

3.2.1 The Back End

The backend of the Curator Interface was coded in Java. All the functionality offered to the curators who interact with the collection or the curator database (see *1.2.4 The Framework* for more information about the components of the Curator Interface) are created using Java Servlets. The use of Java Servlets allows the Website and classes to communicate effectively. Java was selected as the language of choice because the members of the project were already very familiar with it. Furthermore, Java is a platform-independent language, which means that the Bonolo digital repository system can be deployed on any operating system.

The commons fileUpload[1] and commons IO[2] libraries were used to provide an upload function to the curator. All other functions provided to the curators made use of the built-in Java libraries.

[1] http://commons.apache.org/fileupload/
[2] http://commons.apache.org/io/

3.2.2 The Front End

HTML and Cascading Style Sheets (CSS) were used as a means of outputting data to the curator. HTML provides the structures in which the data is presented to the user, while CSS is used to style the HTML and make the interface more user-friendly.

Any development for the front end was done in JSP (Java Server Pages) to allow small sections of Java code to be embedded in the HTML. This allows extra functionality to be added to a Website through the use of Java methods and libraries.

JavaScript and AJAX (Asynchronous JavaScript and XML) were used extensively in the Curator Interface for a wide range of things. JavaScript was primarily used to call pop-up windows prompting the curator for input, as well as calling the necessary Java servlet based on a user's actions. AJAX was then used to dynamically update sections of a Web page depending on the servlet's response. The use of AJAX serves to speed up a Website as it does not have to reload all resources on a page. Various styling adjustments to improve the aesthetics of the Web pages were also done using JavaScript.

The JavaScript library jQuery[1] was used to build the login box in the Curator.

3.2.3 External Components

Apache Tomcat version 7 was chosen as a server application[2]. Tomcat was chosen because it was open source and platform independent. In addition, both members of the Bonolo project had some experience using it to host Java Servlets and JSP pages.

Finally, SQLite was used as a method of creating a database to control the accounts of the curators. SQLite[3] provides a means of creating and interacting with a self-contained, zero-configuration SQL database in a serverless environment. This is ideal for Bonolo as it requires no installation and only requires the sqlite-jdbc.jar to be included in the libraries of the Web application. All interaction with the database is done through Java using SQL.

3.3 Scope and Limitations

The Curator Interface should, ideally, offer all the functions that are available in existing digital repository management systems. However, the time constraints under which this project had to be completed prevented the implementation of a full digital repository management system. Instead, the most essential features required to adequately manage a collection of digital objects were identified and focused on first before any additional features were considered.

The aim of the project was to investigate the effect on usability and performance of using a simple file hierarchy as a data store. While the extra functions offered by other digital repository management

[1] http://jquery.com/
[2] http://tomcat.apache.org/
[3] http://www.sqlite.org/

systems may contribute slightly to the usability of managing collections, the core functionality will be sufficient to assess both the performance and usability of the system.

3.4 Portability and Maintainability

3.4.1 Portability

The initial aim of creating a light weight curator interface has resulted in a system that is easily portable for two reasons. The first is that, with the exception of the Java Runtime Environment[1] (required to run Java applications and applets) and the Apache Tomcat server, all required resources are contained within the Web application. The second is that all the tools and technologies used in the development of the Curator Interface are platform-independent. This in turn has resulted in a system that is platform-independent. The self-containing nature of the Curator Interface, coupled with the tools and technologies used in its construction has resulted in a portable system that can be implemented on a wide variety of systems with very little effort.

3.4.2 Maintainability

The functionality of the Curator Interface has been split into different Java servlets. In addition, the scripts and styles used in the Web site have been organised into separate files for each Web page. This design is modular, allowing individual sections of the Curator Interface to be updated or developed independently. As a result, the entire system is easy to modify, upgrade and maintain.

3.5 Functionality

This section provides details about the features and functions that can be found in the Curator Interface. All the implementation details and issues with each feature will be outlined and explained. The following functionality has been successfully implemented into the Curator Interface.

3.5.1 The Web Interface

The Web interface is the central component between the user and the Curator Interface. All communication that occurs between the users and the repository is done through the front end provided by the Web interface. The purpose of this front end is to provide the curators with a way to use the tools developed for managing collections. Furthermore, the design on the front end was carefully considered so that it could be easily learned and used.

JSP, CSS, JavaScript and AJAX were utilised to facilitate the communication between client and server. The functions that are available to the curators are displayed using JSP and styled using CSS. The actions the curator wants to perform are then captured using JavaScript functions, which call the

[1] http://www.oracle.com/technetwork/java/javase/downloads/index.html

necessary Java servlets or JavaScript methods. The response of the Java servlets was captured using AJAX, which then dynamically updates the HTML to alert the user of the system's response.

The design and implementation of the Web interface took place throughout the development lifecycle. The iterative approach used allowed the Web Interface to improve on previous versions by using the feedback provided at the end of each iteration. As the front end, the Web interface plays a central role in determining the overall usability of the Curator Interface. A large portion of time was, therefore, spent improving the Web interface due to one of the research questions investigating the usability of the system.

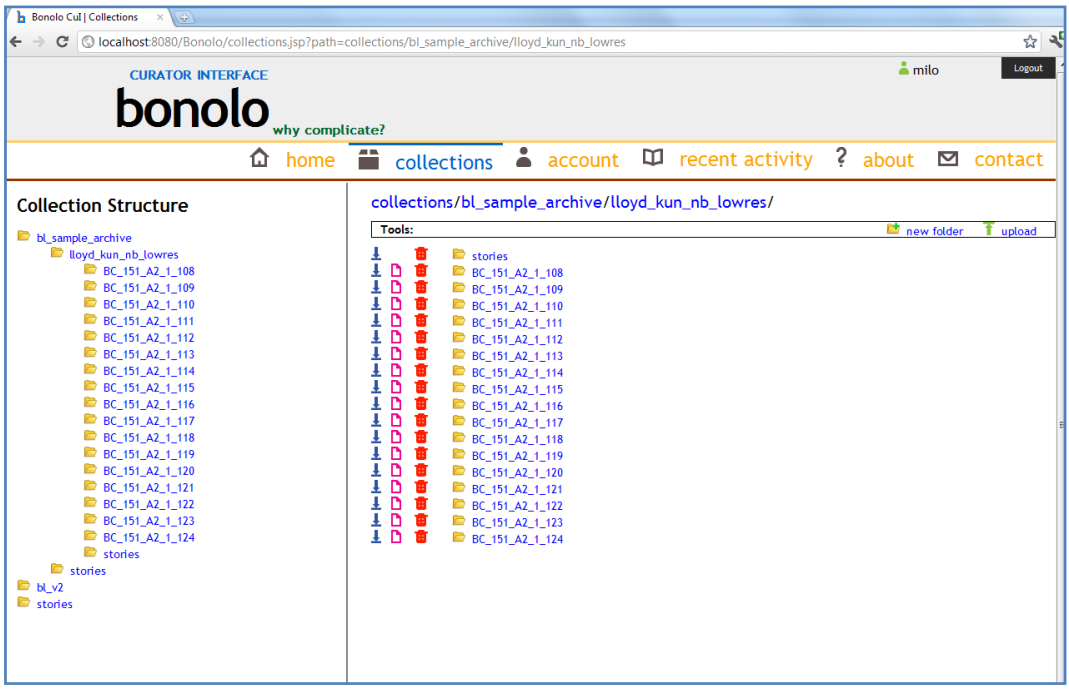

Figure 3.1: Screenshot of the navigation system in the Web Interface

3.5.2 Navigation

The navigation system is perhaps the most essential feature of the Curator Interface as it provides curators with a means of browsing the collections to find the objects they are searching for. The design of the navigation system was modelled on the Windows Explorer interface. In the Windows Explorer interface, the screen is divided into two sections. The left-hand side shows the directory structure, while the right-hand side shows the contents of the current directory. This interface was chosen because of its familiarity among most users, which should reduce the learning required to use the system and therefore improve the overall usability. Navigation of the collections occurs through the collections Web page. See Figure 3.1 for a screenshot of the navigation system.

The initial method of navigation made use extensive use of JavaScript to dynamically create empty HTML div elements on the left-hand side of the screen under each folder. Whenever a user then clicked on a directory on either the left- or right-hand side, the folders within that directory were displayed in the previously empty div element under the directory name on the left hand side. New empty div elements would then be added under each new folder in the list so that the same process

could occur if one of the new folders were clicked on. While only the directory structure was displayed on the left-hand side of the screen, the contents of the current directory were displayed on the right-hand side. The advantage of this approach was that it only needed to get the file list of the current directory. Using this file list, the folders could be added to the appropriate div on the left, while the full content of the file could be displayed on the right. Additionally, whenever a new folder was navigated to, only the browsing area was updated. This results in a smoother browsing experience for the user as the entire page did not have to be reloaded.

However, while this approach did provide some benefits, it did have a significant drawback. It did not allow the curator to go back from viewing an item in the collection to its containing folder. This was due to the dynamic creation of div elements while browsing the collection. By creating the div elements dynamically on click events, it allowed the user to browse effectively, but as soon as the page got reloaded none of the created divs were present any longer. This forced the user to re-navigate through the entire collection every time they pushed the back button on their browser while viewing a file or story. As a result, it was shown in the user evaluation held at the end of iteration 2 that users were frustrated that they had to re-browse through the entire collection each time. This reflected poorly on the usability of the Curator Interface, so another method of navigation was implemented.

The new approach passes a path variable to the collections page in the URL. The path variable contains the path from the root directory to the current directory. The value of the path variable is then processed in the JSP, as well as sent to a Java servlet. The path is used in the JSP to construct the directory structure on the left-hand side of the screen, while it is sent to the servlet return all the contents of the current directory on the right-hand side.

The directory structure is constructed recursively by iterating through all folders contained in the collections directory[1] and adding their names to a list. If the folder name being added to the list is one of the names in the path, that folder is then iterated through in the same manner. This recursive method repeats until the entire directory structure has been completed. This directory structure is then returned and output to the left-hand side of the screen.

The contents of the current directory are shown by getting a list of all its files. These files are then sorted according to length and alphabetical order. The sorted list is then iterated through and all the folders are output. This is done by checking if each file in the sorted list is a directory. If a file is a directory it is printed out, and if it isn't (i.e. if it is a file), the iterator value is added to a vector. Once all the folders have been printed out, the sorted list is gone through again using the values stored in the vector. If there are more than 100 files in the vector, only the first hundred are printed out if they do not end with .zip or .metadata. The total number of files in the directory are then divided by 100 and pages are established for each 100 entries. The only exception to this is if one of the directories in the path is titled "stories". In this case, all the metadata files are printed out as they are assumed to be stories. Stories are metadata files and thus end with a .metadata extension. If an exception is not made for them, they will not be displayed. The structure of the collection that was used had all the story files in the stories folder, therefore any metadata

[1]The collections directory is the root folder for navigation purposes in the Curator Interface, and is the default path when a curator goes to the collections page.

To facilitate paging within a directory, a combination of JavaScript and AJAX is used. Whenever a page link is clicked, a hash tag and page number are appended to the URL. This does not reload the page, but is still a valid address. JavaScript is used to monitor any changes that occur to the URL. If a hash tag is added to the URL, the value of the page number is obtained, and sent with the path to servlet. The results returned by the servlet will be the 100 files following the file (page number – 1) * 100 in the vector. AJAX is once again used to output the results of the servlet to the right-hand side of the screen. Figure 3.2 shows the navigation system on page 2 of a directory.

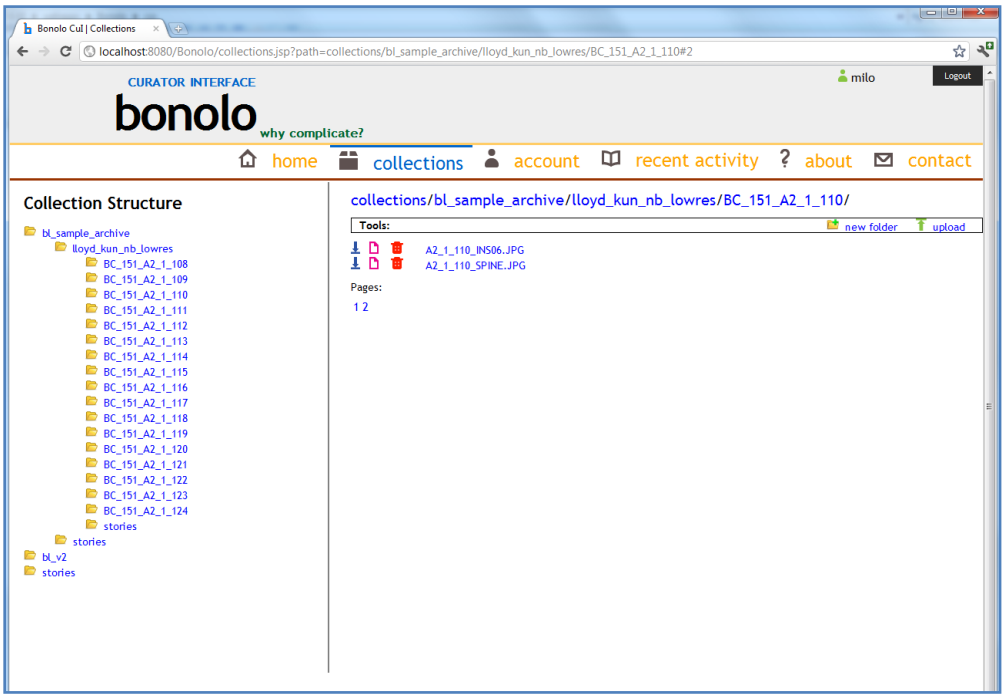

Figure 3.2: Screenshot showing page 2 of a directory in the collection

3.5.3 View Stories

Stories are metadata files that describe sets of digital objects in the collection. These are particularly important to the end user interface, where they are used extensively for searching and browsing. The Curator Interface provides curators with a means of viewing, editing (see *3.4.10 Edit Metadata*) and downloading the story files (see *3.4.9 Download Resources*), as well as uploading a new version of the story (see *3.4.6 Upload Resources*).Viewing the story file calls a servlet that consists of two parts: namely parsing the metadata and displaying the metadata. This approach is also used to view metadata in other areas of the Curator Interface.

Parsing Metadata

Parsing the metadata is done using Java's built in XML parser and begins by obtaining the root element. This is the element that contains all the other tags within an XML file. This root element is then passed to a recursive method which extracts the data stored in each child node. This data is then formatted according to its level in the XML file (i.e. how many levels away from the root it is) and

added to a vector containing all the parsed elements. The pseudocode for this parsing method is provided below.

```
Vector <String> getElements (NodeList nodes, int level) {
        newVector <String>tags;

        FOR each node in nodes where i is the iterator
                IF nodes [i].getNodeName = "#text";
                        // in the case where the node is a text node (i.e. it is not a tag)
                        // do nothing
                ELSE
                        NodeList tagNodes = nodes[i].getChildNodes();

                        IF level == 0 // i.e. the first children of the root element
                                //the first children of the root element are seen to be headings and
                                //are therefore dealt with separately
                                tags.add (level + ":" + nodes[i].getNodeName());

                                IF tagNodes.length == 1
                                        tags.add (nodes[i].getNodeName() + ":" +
                                                tagNodes.item(0).getTextContent());
                        ELSE
                                IF tagNodes.length == 1
                                        tags.add (nodes[i].getNodeName() + ":" +
                                                tagNodes.item(0).getTextContent());

                        tags.add (getElements (tagNodes, level + 1));
        END LOOP
}
```

Displaying Metadata

The metadata is displayed using a table format with the tag name on the left and the content on the right. This table is constructed using the parsed metadata. This table is then passed back from the servlet as HTML. AJAX was used to invoke the servlet using an XMLHttpRequest[1]. The response of the servlet was then captured in the AJAX method, and embedded into the Web page.

3.5.4 View Files

Curators are able to view the resources in the collection. This is necessary as they will need to know what the resources look like in order to organise them effectively. The path to the digital object is submitted via a hidden form field. This information is extracted with the use of Java code in the JSP,

[1] http://www.w3schools.com/xml/xml_http.asp

and the appropriate object is called from the collection. This object is then formatted using CSS to fit on the curator'ss screen. Figure 3.4 shows how files are displayed in the Curator Interface.

A file's associated metadata is displayed alongside it. The approach used to display this metadata has been explained in *section 3.4.3 View Stories*.

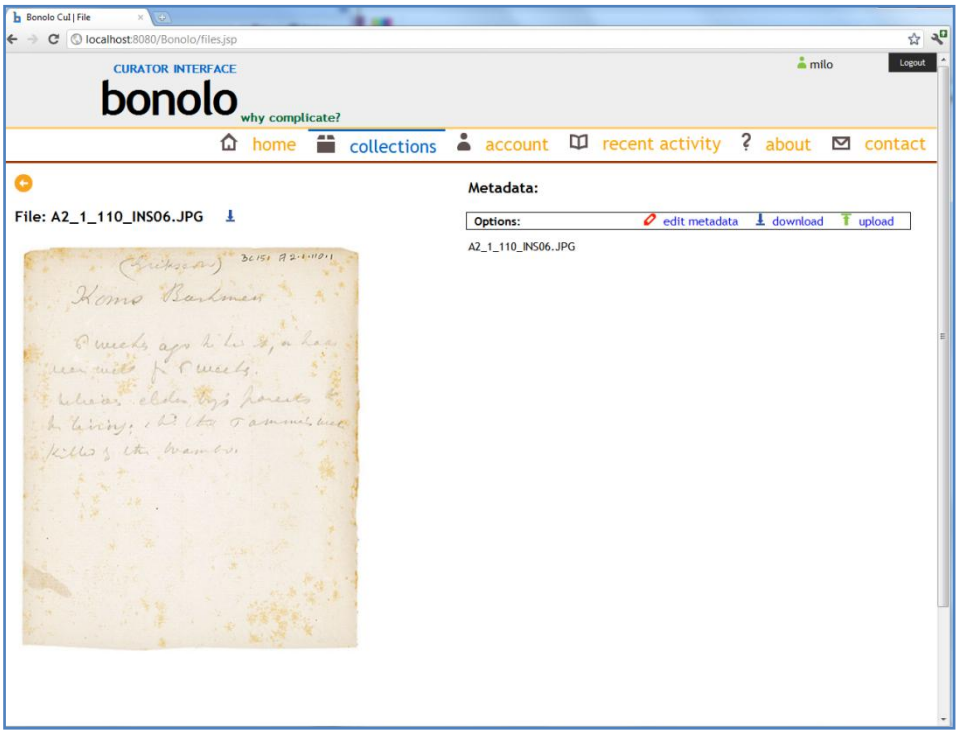

Figure 3.3 Viewing files in the Curator Interface

3.5.5 Upload Resources

Uploading resources involves providing curators with the means to add new items to collections. The standard Java libraries do not support the uploading of files to a Web server. For this reason, the Commons FileUpload [Apache Commons, 2010] and Commons IO [Apache Commons, 2011] libraries were used. These libraries extend Java to allow file uploading between a client and a Web server. The upload process begins when a curator clicks on the upload button, prompting them to choose a file on their computer. Once a file has been chosen, and the upload file button has been pressed, the file is uploaded to the server where it is extracted if necessary, and metadata is updated where possible. Each of these three steps are broken down below. A different method was used for uploading single files in the file, story and metadata pages in the Curator Interface. This was done because only single metadata files should be uploaded on these Web pages.Using this approach, the file is only uploaded to the server, as extraction and updating the metadata should not be required.

Upload

When the upload is confirmed, the file is sent in a form along with the position in the repository to a Java servlet. The position is saved as a string on the servlet, while the file is saved as a FileItem. A new file is then created on the server at the appropriate position and the conents of the FileItem are copied from the user's computer to the server.

Extraction

Initially, the upload function did not support batch uploads as folders cannot be selected for upload by the browser. The upload function was then extended to allow users to upload a batch of files and folders to the servlet by zipping them together into a single ZIP file. Once the ZIP file has been uploaded, its contents are extracted using Java's built-in ZIP library. This did not require a recursive method as all the entries in a ZIP file can be read using the zipFile.entries() method.

Each entry in the ZIP file was processed sequentially. If the current entry was a folder, a new folder was created in the collection.If the current entry was a file, that file was copied out of the ZIP file to the appropriate location. This process was followed until all the entries in the ZIP file had been extracted.

Update metadata

The final action that occurs when an upload action takes place is metadata updating. While all files in the collections should have metadata associated with them, it is not a requirement for a folder to have metadata. However, some folders do have a corresponding metadata file. These folder metadata files usually contain information about the actual contents of the folder. When changes are made within a folder, the contents of the folder's associated metadata file should also be updated.

The structure of the metadata files is not known, so the following assumptions about the folder metadata were made:

- The folder metadata file only contained information about its contents
- Each file in the folder had an associated tag in the metadata file
- The file tags would be children of the root element.

Using these assumptions, the name of the first tag that contained text was recorded. An empty child element was then created with the value of the tag name. Following that, a text node was created that contained the name of the uploaded file. This text node was then appended to the child element, which was subsequently appended to the root. The result of this process is that a new tag containing the name of the uploaded file is added to the root.

For example: Figure 3.4 shows the metadata associated with the lloyd_kun_nb_lowres folder. The root element in this case is "collection", while the children of the root element are "book". Using the assumptions made, the lloyd_kun_nb_lowres folder will only contain the elements listed in the metadata as "book" (refer to Figure 3.1 to verify this). If a new folder were to be added to the lloyd_kun_nb_lowres folder, the following would be appended to the "collection" element: <book>newFolder</book>.

Finally, the metadata file has to be updated to show the changes. This is done using a TransformerFactory, which is part of the Java XML library. This TransformerFactory was used to create a string from the XML data contained in the root. This string was then written out to the metadata file and by doing so, the metadata was updated.

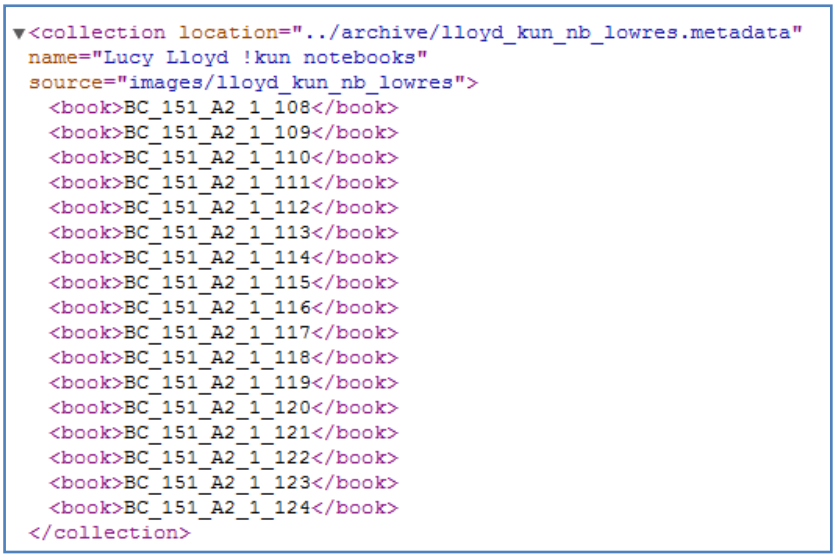

Figure 3.4: The assumed metadata structure of folders

3.5.6 Create Folder

A create folder tool has been developed that allows curators to create new folder in the collections. File creation is supported on the collections Web page, where users can browse through the file structure. This is done by prompting the user to enter the name of the folder they wish to create. The input supplied by the user, as well as the position in the collection are sent to a servlet. This servlet then creates a folder with the desired filename in the correct location. If a folder with the same name already exists at that location, the user will be alerted and prompted for another name.

If the directory in which the folder was created has an associated metadata file, it will be updated to reflect the changes. This occurs in the same way as explained in *section 3.4.5 Upload Resources*.

3.5.7 Delete Resource

Any objects in the digital repository can be deleted. This allows curators to remove objects that no longer belong in the collections. File creation and delete functionality are available through the collections Web page. When a user chooses to delete an object (a single file or an entire folder) from a collection, they are prompted to confirm their actions. This is done to prevent any objects from being deleted by accident. Once the user has confirmed the delete action, the path to the file is submitted to a servlet where the delete action is processed and metadata is updated where necessary.

Delete

The deletion of files is done recursively. This ensures that if a folder is deleted, all files it contains are deleted as well. If a single file is to be deleted, no recursion will occur and the file will simply be deleted. However, if a folder is to be deleted, the program will first go through the contents of the folder and delete all the individual files and folders within. Should another folder be contained within the original folder, the same process will occur for that folder. Finally, once all the files within the original folder have been deleted, the folder is deleted.

Update metadata

As in the case of uploading resources and creating new folders, the built-in XML parser of Java was used and the same assumptions were made about the metadata structure. However, in the case of resources being deleted, certain elements had to be removed from the metadata instead of appended. To do this, the children of the root element were searched sequentially. If one of them was found to have the same value as the name of the file being removed, that child node was removed from the root. Once the node has been removed, the XML is converted to a string and printed out to the metadata file, updating it.

3.5.8 Download Resources

Curators have the ability to download elements from the collection. When a download call is invoked by a curator, a Java servlet is called to perform the download process. This process is done in two separate parts, namely compression and transferral. Each of these is discussed in more detail below.

Compression

If the curator chose to download a folder from the collection, the first step in the downloading sequence is compression, which is done for two reasons. The first is that it reduces the size of the files being transferred, therefore reducing the amount of data to be transferred as well as the time taken for the download to complete. The second is that it allows the curator to download entire folders from the Curator Interface.

Compression was done using Java's built-in ZIP library. When the user decided to download a folder, a ZIP file was created in the collection. The folder that the user wishes to download is then output into the ZIP file using Java's ZipOutputStream. A recursive method was used to add the contents of the root folder, and any sub folders to the ZIP file.

Transferral

Once the ZIP file has been created with the necessary contents, it is transferred to the user. This is done by setting the response type of the servlet to a bit stream that is interpreted by the user's computer as a download. The contents of the ZIP file are then read by the servlet, and written out to the client over the bitstream. Once the file has been successfully transferred, the ZIP file is deleted. If any problems with the download occur, causing the download to be cancelled, the ZIP file will be deleted so it does not take up extra space on the server.

In the case that the curator wants to download a single file (for example an image or metadata file), the compression section is ignored and only transferral occurs. The only difference in the transferral procedure is that the file is not deleted on completion.

3.5.9 Edit Metadata

The ability to edit metadata is essential to the curators of a collection, as the metadata describes the digital objects contained in the collection. Any changes or edits that need to be made to the digital objects are done through editing the metadata associated with them. The original approach to allow curators to edit metadata involved parsing and displaying the metadata in a manner that would allow users to edit specific elements directly. However, the complex nature of XML files provided issues when designing an interface to allow users to edit the metadata. This approach was abandoned in favour of the following approach as time was becoming an issue and there were other areas of the Curator Interface that still required development. Figure 3.5 shows the metadata editor.

Curators are able to edit the metadata by viewing the plain text in a popup window. The contents of the metadata file are retrieved from the file using XMLHttpRequest, with a plain text response. The response is then added to a text area in a popup window, allowing the user to make the necessary changes to the document. Once the changes have been made, the content of the text area are sent to a servlet. This servlet then overwrites the old metadata file with content from the text area.

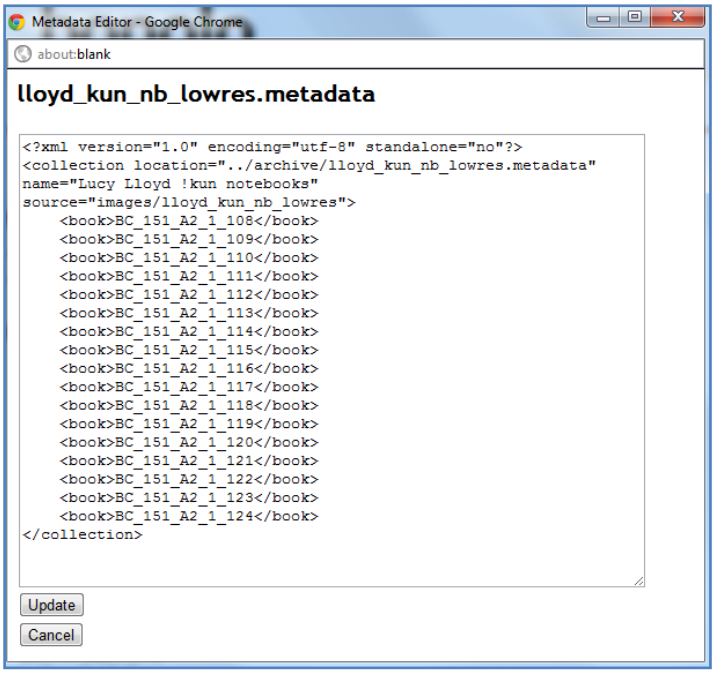

Figure 3.5: Curator Interface Metadata Editor

3.5.10 Curator Accounts

To impose some degree of access control on the collections, curator accounts were added to the Curator Interface. The primary function of these accounts is to determine which collections a curator

can modify. For the purposes of storing the curators' details, a SQLite database was used. The database contained three tables: a Users table; a Collections table; and a Waiting table. The function and structure of each of these tables is outlined below:

- **Users Table**
  - Contains details about the users who have been registered on the Curator Interface.
  - The fields in this table are:
    - Username
    - Password
    - E-mail address
- **Collections**
  - Contains details about which collections are available to which users.
  - The fields in this table are:
    - Collection
    - Username
- **Waiting**
  - Contains details regarding which users are requesting curatorship and the collections they are requesting to curate.
  - The fields in this table are:
    - Username
    - Collection

Through using this database, Register, Login, Logout, Access Control and External Changes are all made possible. See Figure 3.6 (end of *section 3.5*) to view the user account page.

3.5.11 Register

New users will be required to register before they have any degree of control. When registering, the users will have to enter a username, password and e-mail address. The username and e-mail address must be unique if they are not, notification will be given to the user to inform them that the username or email has already been registered. The users will also have to select the collections they wish to curate from a list of available collections on the server. Once the user has provided the necessary information, they will be registered on the Curator Interface with their username, password and e-mail address being added to the Users table. The username and collections the user has requested to curate will be added to the Waiting table. This is all done through a Java servlet that uses the SQLite JDBC to connect and update the database. Prepared statements are used to update the database while traditional SQL queries are used to get information. Permission to curate a collection must be given by one of the existing curators (for example, if Curator A requests permission to curate collection X, one of the existing curators of collection X must approve A's request). Each user can see the requests of others to curate collections they are in charge of on their account page, where they are able to

approve or decline requests. Approving will move a user from the Waiting to the Collections table, while declining will just remove a user from the Waiting table.

3.5.12 Login

Users will have to login before they have access to the tools provided by the Curator Interface. This is done through the login box located at the top right of every Web page. The users will have to provide their e-mail address and password in order to be successfully logged on. If either the email or password is incorrect the user will be notified. This too is done through a Java servlet with the SQLite JDBC. Requests are made to the Users table to confirm that the email address and password entered by the user are valid and correspond to a single user. A session variable will then be set with the username to show that the user has successfully logged in. The username of the curator will be displayed at the top right of the screen, and the login box will change to a logout box. To confirm that the user is logged in, on every page load, the username session variable is checked. If there is no session variable, it is assumed that there is no user logged in, and therefore no collection altering functionality will be available.

3.5.13 Logout

The logout function calls a Java servlet that removes the username session variable. This will mean that there is no logged in user, and all functionality will be removed.

3.5.14 Access control

The session variable containing the username is used extensively to determine what actions a curator can perform. The navigation servlet makes use of the username to determine what collections it can allow the curator to view. No content will be shown in collections that the user does not have curatorship over. This is done by querying the Collections table and checking that the user is a curator of the current collection. Furthermore, users will not have the ability to upload and create folders in collections they do not have curatorship over.

Any folders that are uploaded to or created in the collections directory are assumed to be independent collections. Users who add collections to the repository will automatically be given curatorship over them. This is done by checking the location a folder is being created at to see whether it is the collections directory. If it is, the collections directory, the collection and username are added to the Collections table. Files should not be uploaded to the collection directory as a single file is not considered an entire collection, and will therefore not be displayed. This file will be deleted as it is wasting space on the server.

Similarly, if a collection is deleted all entries from the Waiting and Collection tables will be removed. This is done by executing an update on the Waiting and Collection tables that removes items where the Collection field equals the name of the collection that was just deleted.

3.5.15 External Changes

The first user to register on the Curator Interface is assumed to be the administrator. This does not provide any extra functionality, but they will be given curatorship over any collections that are added directly to the server (i.e. collections that were added to the server, but not through the curator interface). A check is done every time the collections page is visited to see if there are any discrepancies between the number of collections on the server and the number of collections in the Collection table. If any differences do exist, the database will be updated to reflect the changes in the file structure. If collections have been deleted, the Collections and Waiting tables would be adjusted by removing entries. If collections have been added, the Collection table will be updated to include the new collection with the first user in the User table set as the curator. The only time this would occur is when changes have been made to the collections directory outside the Curator Interface.

3.5.16 Recent Activity

The recent activity feature allows users to view the changes that have occurred within the collection. Viewing the recent activity that has taken place in the collection is available in three separate locations on the Curator Interface. Each of these serves a different purpose, which is explained under *Displaying Changes*. In order for the recent activity feature to be implemented, two actions must be performed. These actions are: recording changes to the collection and displaying changes.

Recording Changes

The actions that modify the collection are as follows: uploading a resource to the collection; deleting a resource from the collection; creating a folder; and editing metadata. Whenever one of these is performed, it needs to be documented so that it may be displayed later. For the purposes of recording the changes made to the collection, an XML activities document, which is stored on the server, has been used.

Any change that is made to the collection will update the activities document. However, this will not be done in the same manner that uploading and creating a new folder uses to update existing metadata. The rigid structure of this XML document allows a more efficient method to be implemented using RandomAccessFile[1]. RandomAccessFile has a method that returns the length of a file, as well as a method that sets the length of a file. Using these tools and knowledge of the document's structure, the length of the file was reduced so as to delete the container's closing tag. Once this closing tag was deleted, the new child was printed out to the document, followed by the container's closing tag. As a result, a new child node was appended to the end of the document.

This method was seen as more efficient as it will take the same time regardless of the size of the file, while the other method (see *section 3.4.5 Uploading*) has to reconstruct the entire document, therefore taking longer each time as the length of the document increases. This is an important factor as the activities document is expected to get quite large as more modifications are made to the collections.

[1] http://download.oracle.com/javase/1.4.2/docs/api/java/io/RandomAccessFile.html

Displaying Changes

AJAX was used to read the contents of the activities document using XMLHttpRequest. The XMLHttpRequest reads the document on the server and then responds by returning the data stored in the document. ResponseXML ensures that the data from the XMLHttpRequest is XML data. This XML data is easily parsed as the tag names of the XML document are known. The parsed XML data is then formatted and displayed.

As previously mentioned, there are three seperate locations where the actions of the curators are displayed. The first is the home page. The home page shows the ten most recent actions that have occured. This allows the user to see which areas of the collection are being worked on while they are logging in to the system. The second is the user's account page. The account page shows the ten most recent actions the logged in user has performed. This allows the user to remember what areas of the collection they were working on. Links to the areas where these actions took place are provided for easy navigation. The third and final location is the recent activity page. This page shows all the actions that have taken place. It allows the user to filter by action type, and to sort by most recent or oldest. This filter adjusts the restrictions placed on the parsed XML data to only show the selected action, while the sort changes the direction the response data is traversed. Figure 3.6 shows the recent activity on the user account page.

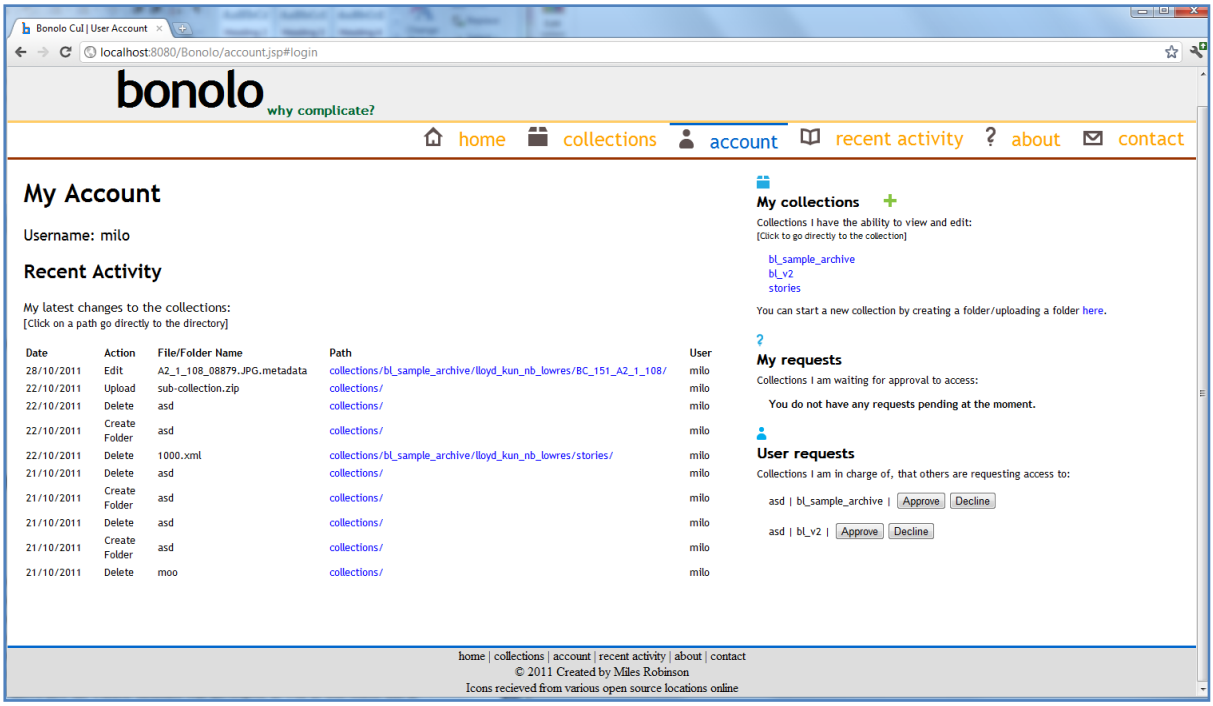

Figure 3.6: User account page in the curator interface

3.6 Iterations

The development of the Curator Interface took place over three iterations. Each iteration consisted of time allocated to the design, implementation and evaluation of the system. This approach was adopted

as it allowed the system to improve on areas of weakness from the previous iteration while adding more features and functionality. The goal of the first iteration was to prove the concept could be successfully implemented. The second iteration sought to develop an initial implementation of the Curator Interface, with the third iteration resulting in a completed system. Each iteration is discussed in more detail below. Figure 3.7 shows how the Curator Interface was developed, as well as deliverable due at the end of each iteration.

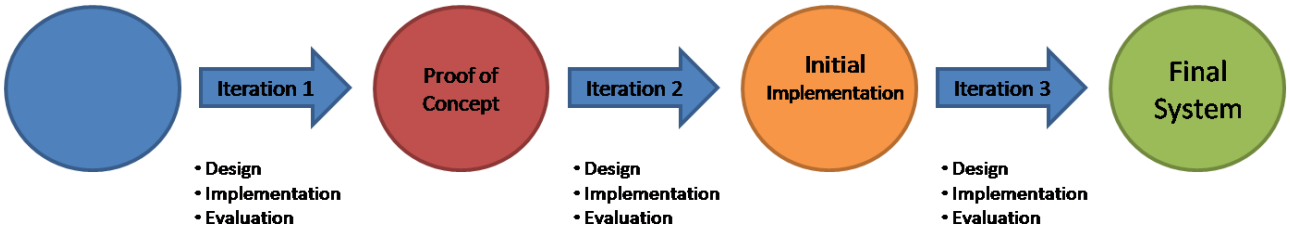

Figure 3.7: The iterative approach used to develop the Curator Interface

3.6.1 Iteration 1 (15/06/2011 – 15/07/2011)

The objective of the first iteration was to demonstrate the technical feasibility of constructing a Curator Interface to manipulate a collection that consisted of a file hierarchy. For the purposes of this iteration, navigation was identified as the core function to be implemented.

In Iteration 1, a significant portion of time was spent learning the tools and technologies that would be required for the remainder of the project. This was in addition to the time spent on the design, implementation and evaluation phases.

Design

In the design phase of this section the core functionality of the Curator Interface was researched and discussed. Additionally, the feature that would be implemented during the first implementation was decided upon. The navigation feature was seen as the most important because the curator must first be able to view the files in a collection before they can be effectively managed. For this reason, it was decided that navigation would be implemented as it would also demonstrate the ability to deal with a file hierarchy.

Implementation

The following changes were made to the Curator Interface during Iteration 1:

- **Web Interface**
    - A basic Website was created to demonstrate the navigation capabilities of the Curator Interface.

- **Navigation**
    - The basics of the navigation functions were implemented, demonstrating the capability of the system to address and interact with a simple file hierarchy.
        - The navigation feature made use of AJAX calls to dynamically update and create *div* elements around the Website.
- **Viewing Files**
    - Image viewing is supported, but not stable.

Evaluation

The evaluation phase of Iteration 1 was the shortest, as very little functionality was available to be evaluated. The evaluation method that was used during this phase was primarily self-evaluation, with a focus on the look and feel of the Website. The project supervisor and second reader (Dr. Anne Kayem) provided limited feedback at this point. It was noted during this phase that improvements in the presentation of the navigation could be made in order to ensure that there is no confusion amongst the users. The nature of the presentation was, however, expected as this iteration's objective was to prove the concept was possible.

3.6.2 Iteration 2 (16/07/2011 – 14/09/2011)

The second iteration's objective was to produce an initial implementation of the Curator Interface that supported the majority of the functionality to be included in the final version. During this iteration, the majority of the functionality offered by the final system was developed and added to the Curator Interface.

Design

During the beginning of the design phase of iteration two, a focus group was held. The focus group consisted of two overseers and six members of the Computer Science Department of the University of Cape Town. Four of the members that took part were Masters Students and members of the Digital Libraries group within the Department. The remaining two members were honours students with some Web development experience. The purpose of the focus group was to determine what tools should be included in the Curator Interface.

The information gathered from the focus group was then analysed and refined to determine which features were essential components of the Curator Interface. The following was obtained:

- **Core Functionality:**
    - Navigate through the collection
    - Add objects to the collection
    - Delete objects from the collection
    - Edit objects currently in the collection

        - Note: The editing of an object is done through making changes to its associated metadata file.

- **Additional Functionality:**
    - Batch uploads of resources
    - Export a collection to a different format
    - Download objects from the collection
    - Provide some statistical feedback
    - Highlight the changes that were made to the collections.
    - Provide a basic search
    - Control the access of curators to collections
    - Rapid editing of metadata through an offline spreadsheet paradigm.

Due to the time constraints of the project, the core functionality was implemented before any additional functionality. This was done to ensure that the curator interface had sufficient functionality to adequately manage a collection before it was evaluated.

Implementation

The following gives a high level description of the changes that were made to the Curator Interface during the second Iteration:

- **Web Interface**
    - These Web pages were added:
        - A home page showing some of the recent changes that have been made to the collections.
        - A collections page that allows the curator to browse and manage the collections stored in the digital repository.
        - A recent activity page showing the changes that have been made to the collections.
        - An about page that provides some information about Bonolo and its two components, the user and curator interfaces.
        - A contact page that allows users to raise concerns or suggestions with the developers.
    - Toolbars showing the functions available to the curator were added
    - The CSS was edited to enhance the aesthetics
- **Navigation**
    - The navigation window was modified so that it uses side by side panels. This was done to provide a similar interface to that of Windows Explorer.

        - The left panel shows the directory structure of the collection so the curator can quickly access another area of the collection.
        - The right panel shows the contents of the current directory.
- **Upload**
    - Upload functionality was added to the system to allow curators to add single files to the collection.
- **Create Folder**
    - Curators were provided with a means of creating new folders. This allows them to start new collections on the server, as well as organise the contents within the folders.
- **Delete**
    - A delete function was added to the system that allows curators to remove items from the collection.
- **Recent Activity**
    - All of the recent changes made to the collection can be viewed and filtered.
- **View Stories**
    - The contents of the storyname.metadata files can be viewed.
- **View Files**
    - The images and their associated metadata can be viewed.

Evaluation

Brief user evaluations took place at the end of Iteration 2. The purpose of these evaluations was to get feedback from the users as to how the system could be improved for the final iteration. For the purpose of the Curator Interface, user evaluations were performed with nine individuals who had no prior experience in using the system. Users with no experience were chosen because their feedback would be unaffected by any previous learning that may have occurred, and would thus be the most valuable in determining the intuitive usability of the system. Nine people from different faculties within the University of Cape Town performed the user evaluations and provided feedback.

The evaluation, following the constructive interaction method [Kahler et al., 2000], required the users to perform a set of tasks while being observed. After the user completed the set of tasks, they were asked about their experience as well as any surprising actions that they performed. Finally, they were asked about any improvements and additional features that should be added to the system. The significant results of the evaluation are summarised here:

- All 9 users were able to complete all the tasks in the evaluation.
- No users had any issues with creating a new folder, uploading a resource, or deleting a resource.
- All the users expressed some degree of frustration regarding the navigation system, as they could not use the back button on their browser as an additional navigation tool.

- 3 users pointed out that it would be useful to include a back button on the Web pages that displayed the images and stories to return to the containing folder.
- 5 out of the 9 users said that the overall usability was of an acceptable standard, with the remaining 4stating that the usability was below average due to the navigation.
- 1 user pointed out that it would be useful to see which files have metadata associated with them while browsing the collection, as well as which files do not.
- 4 users suggested that icons be used around the Website to improve the look and feel.
- It was noted by 2 users that the information about the actions performed on the collection should include the username of the curator who executed the actions.
- 1 user suggested that the "sort by" options in the filter of recent activity should be changed to "most recent" and "oldest".
- 7 of the 9 users suggested that a confirmation window be brought up when an item is deleted from the collection.

3.6.3 Iteration 3 (14/09/2011 – 12/10/2011)

The third and final iteration aimed to produce a final digital repository system with sufficient functionality to allow curators to adequately manage a collection. During this iteration, modifications to the core functionality were made. Additionally, some auxiliary functionality was added to the Curator Interface to improve the user experience while curating collections. As this was the final iteration in the software development life-cycle, a large amount of time was spent performing evaluations of the system.

Design

The results of the user evaluations that took place at the end of the second iteration were used in determining what changes had to be made to the prototype to construct a completed system. In addition to the user feedback, the results of the focus group were used to determine what features to add. The following list details the user feedback for changes that need to be made to the initial implementation.

- **Web Interface**
  - Add icons to illustrate functions.
- **Navigation**
  - Needs to allow the back button on the browser to take the user to the previously viewed directory.
  - Add buttons into the file and story pages that allow the user to go to the containing folder.
  - Give some indication to the curator about which files and folders in the collection have metadata associated with them and, more importantly, which do not.

- **Recent Activity**
    - Include the username of the curator who performed an action on the collection.
- **Delete**
    - Add a confirmation dialog to the delete function.

Implementation

All of the user suggestions were implemented in the final iteration. The following list shows the changes that were made to the various functionality of the Curator Interface during the third iteration excluding those suggested by the users:

- **Web Interface**
    - A login/register box was added to the top right of each Webpage.
    - The following Web pages were added:
        - A registration page that allows new curators to join, and apply for permission to curate collections stored in the repository.
        - An account page that shows:
            - The collections a user has curatorship over
            - The collections a user is applying for permission to curate
            - Requests from other users to curate collections the current user has curatorship over
            - The recent activity of the user
    - A metadata page that allows the curators to edit the metadata of objects in the collection.
- **Navigation**
    - Iteration 2's user suggestions were implemented.
- **Upload**
    - The upload function was developed further to allow for batch uploads of files.
    - Uploading directories to the collections directory added a new collection to the curator database. The user who uploaded the directory was given curator rights.
- **Create Folder**
    - Creating a folder in the collections directory added a new collection to the curator database. The user who created the folder was given curator rights.
- **Delete**
    - Deleting a collection will remove the collection from the database.
- **Recent Activity**
    - Recent activity of each curator is available in the user's account.
    - The username of the curator who performed an action was added.
    - A filter by 'Edit' option was added to the recent activities page.
- **View Stories**

    - A more general method of parsing and displaying the information in storyname.metadata was developed.
- **Edit Metadata**
    - A metadata editor pop-up was added allowing the curators to edit file, story and folder metadata. This tool was made available on the file, story and metadata pages.
- **Curator Accounts**
    - Curator accounts were added to the system. These accounts determined which users had the ability to curate certain collections.

Evaluation

The final evaluation that took place consisted of both user and performance evaluation. These evaluations were done to determine the effects a simple file hierarchy has on the usability and performance of the Curator Interface. As the results of these evaluations were used to answer the research questions, they have been discussed and analysed in more detail in *section 4. Evaluation.*

3.7 Issues

The following issues were not solved during the design and implementation of the Curator Interface:

- Parsing the metadata files
    - The current method of parsing the XML metadata files works fine using the sample collection provided, however, more complex XML structures are not parsed as well.
    - A more generic solution needs to be implemented.
- Race conditions
    - If two users modify the same document at the same time, only the version of the user who saves last will be committed to the collection. Some flag or mutex system needs to be implemented so that only a single user can edit a document at a time.
- Popup windows
    - Popup windows provide a means of dialog between the Curator Interface and the user. If the user's pop-up blocker is enabled, this dialog cannot occur.

3.8 Discussion

This chapter has described the design and implementation of the Curator Interface. The choice of the tools and technologies used in the creation of the Curator Interface has been outlined and justified due to the platform-independent nature of the tools. The functionality supported by the final system consists of the core features along with some additional features. The core features were implemented first due to time constraints and the requirement of a functioning system for evaluation purposes. The design and implementation of the Curator Interface followed an iterative approach, allowing development of the system to build on the evaluations and feedback of the previous iteration. The final system is portable and maintainable due to its platform-independent nature along with its modular design. [There were issues with implementation.] Using the information provided in this chapter, a Curator Interface that produces similar results as those found in the *Chapter 4. Evaluation* of this report can be created.

Chapter 4

Evaluation

4.1 Introduction

The purpose of evaluation is to determine the overall success of the Curator Interface. For the Curator Interface to be deemed a success, the following criteria should be met:

1. Using a simple file hierarchy as a file store does not affect the overall usability of the Curator Interface.
2. The performance of the Curator Interface is not be hindered by using a hierarchical file based data store.
3. The Curator Interface is comparable to other digital repository management systems.

To thoroughly answer these questions, both user and performance evaluation were performed.

4.2 User Evaluation

4.2.1 Aim

The aim of the user evaluation was to get feedback on how users felt about the experience provided by the Curator Interface when managing collections. This was done in order to answer research questions one and three:

- Will using a simple file hierarchy as a file store affect the overall usability of the Curator Interface?
- Is the Bonolo Curator Interface comparable to those of other digital repository systems?

4.2.2 Methodology

- Twenty users took part in the user evaluation. These users did not need to have any prior experience in using a digital repository system.
- Each evaluation was performed on a Local Area Network.
- The browser used in each evaluation was either Google Chrome 14.0.835 or Mozilla Firefox 7.0.1 depending on the user's preference.

- Any changes made to the collection during an evaluation were reset so that the collection was in its original state before the following one began.
- The collection used for the evaluations had the following attributes:
    - The collection held seventeen books.
    - 1460 images and their associated metadata (2920 files in total) were available to be accessed through the books.
    - Each book contained between 22 and 102 images and their associated metadata files.
    - The collection held a directory with 186 story metadata files.
- A handout (see *Appendix B*) was presented to each user. This handout contained:
    - A consent form, allowing the results of the evaluation to be used in this report.
    - A set of tasks that the user was required to perform.
    - Questions about each set of tasks.
- The set of tasks in the handout was designed to expose the users to all the functions required to manage a collection, as well as allow them to explore how the Curator Interface functions.
- At the end of each task, the user answered a set of questions that was designed to obtain feedback about the experience of the user. The questions were made up of:
    - Closed questions that required the user to choose an answer on a scale of 1 to 5. While many of these questions had differently worded answer scales, in each case 1 was the equivalent of very negative, while 5 was the equivalent of very positive.
    - Two yes/no questions to gain more insight into the user's experience.
    - Open questions where users could express any specific frustrations and provide suggestions.
- The results of the questions were then compiled and analysed to answer the appropriate research questions.

4.2.3 Results

The results of the yes/no questions are shown in Table 4.1, while Table 4.2 shows the results of the closed questions requiring the users to answer on a scale of 1 to 5. The written responses have not been included in this report. However, they are used in *4.2.4 Analysis* to better understand the results obtained from the closed questions.

Table 4.1: Table showing the results of the yes/no questions in the handout

		Response	
	Question	Yes	No
Task 6	b) Have you ever worked with XML documents before?	8	12
Post Task Questions	9) Have you ever used a Website allowing you to manage data stored on a server?	20	0

Table 4.2: Table showing the results of the closed questions in the handout

		Responses				
	Question	5	4	3	2	1
Task 1: Registration	a	14	4	2	0	0
	b	18	2	0	0	0
	c	19	1	0	0	0
	d	12	8	0	0	0
Task 2: Navigation and Viewing Images	a	11	5	4	0	0
	b	12	5	3	0	0
	c	11	7	2	0	0
Task 3: Create and Control Access to a Collection	a	13	3	4	0	0
	b	11	6	2	0	0
	c	13	7	0	0	0
	d	12	8	0	0	0
Task 4: Uploading	a	15	5	0	0	0
	b	13	7	0	0	0
	c	10	9	1	0	0
	d	14	6	0	0	0
	e	15	5	0	0	0
Task 5: Downloading	a	11	8	0	1	0
	b	13	4	3	0	0
	c	13	6	1	0	0
Task 6: Stories and Editing	a	2	6	10	2	0
	b	0	8	9	3	0
Task 7: Deleting	a	18	2	0	0	0
	b	15	5	0	0	0
Task 8: Flexibility	a	14	6	2	0	0
Post Task Questions	1	8	12	0	0	0
	2	9	11	0	0	0
	3	6	13	1	0	0
	5	12	8	0	0	0
	6	14	5	1	0	0
	8	5	12	3	0	0

4.2.4 Analysis

Table 4.2 shows that in general users did not have problems performing tasks 1; 4; 7; and 8 as the results received were positive, indicating a high level of usability in those areas. Some users did,

however, experience difficulty performing tasks 2; 3; 5; and 6. The analysis of the user evaluation will focus on the tasks users had difficulty with, along with the results of the post task questions.

Task 2: Navigating and Viewing Images

Although Table 4.2 indicates that some users found the navigation and viewing files slightly less effective than some of the other tasks, the response from the majority was positive. The users who experienced this issue stated that they did not have a problem with the manner in which navigation was done, but rather that they would like to have access to a search function. There is, therefore, not an issue with the current navigation system, but the usability could be improved through the inclusion of a search function.

Task 3: Create and Control Access to a Collection

As in task 2, it can be seen from Table 4.2 that the majority of the responses to this task were positive. The users who did experience some difficulty stated that they struggled to find where to create a file as opposed to the actual process of creating a file. It was suggested in the written section of this task that the location for creating a new collection be made more obvious to the user. This shows that there is not an issue with the tools, but a lack of information provided to inexperienced users. This was highlighted in question c) of task 3, which asked if the user would find the process of creating a new collection easier the second time round. All responses to this question were positive showing that the user's ability to perform this task would improve with experience.

Task 5: Downloading

Table 4.2 shows that a single user had difficulty with locating where to download a file. This can be dismissed as an outlier result, as the remainder of the users did not experience any difficulty when performing this task. It was this same user that rated the overall download experience as a three, whereas all other responses to the overall download experience were positive. The user explicitly stated that the lower rating was due to the difficulty in locating where to download a file, and suggested that the icon be made slightly larger.

Task 6: Stories and Editing

Task 6 was met with the most difficulty amongst the users. It was expected that users without experience in working with XML documents would have difficulty with this task. For this reason, users were specifically asked if they had worked with XML documents before, so that a more informed result could be obtained. Figures 4.1 and 4.2 illustrate the differences between those users with XML experience and those without.

It can be seen when comparing Figure 4.1 and Figure 4.2 that users without XML experience found the editing of stories to be more ineffective than those users with XML experience. This was highlighted in the written section of the task, as users without XML experience stated that their lack of knowledge made it difficult to figure out how to proceed. It should be noted, however, that all users who took part in the evaluation were able to successfully edit the document after analysing the structure of the XML document. It is reasonable to assume that the majority of curators will have experience working with metadata stored in an XML format, and will thus have a similar response to

those users with XML experience. However, it can be seen from Table 4.2 that the usability of this section, while still slightly positive, is not as good as the other sections and can, therefore, be improved.

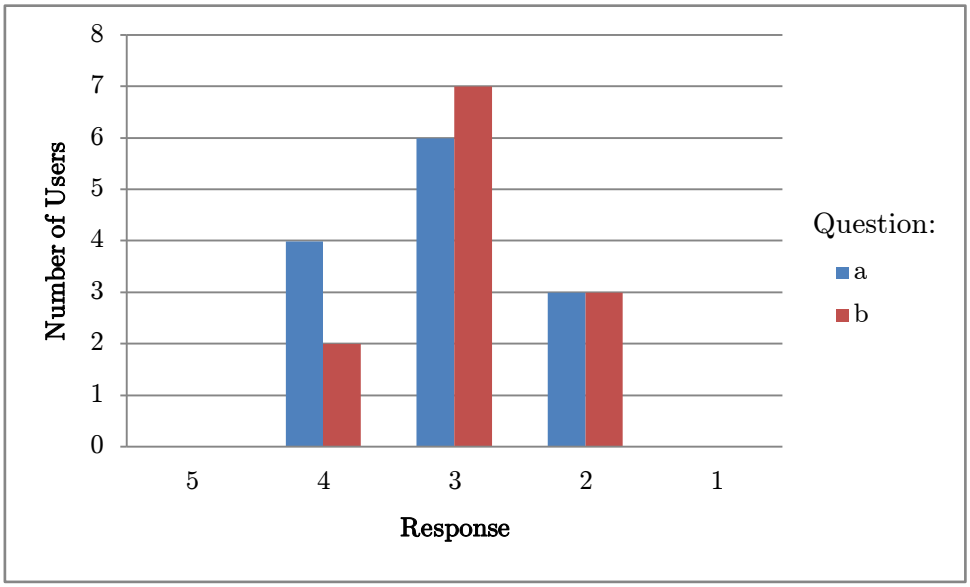

Figure 4.1: Graph showing the responses of users with no XML experience when completing task 6

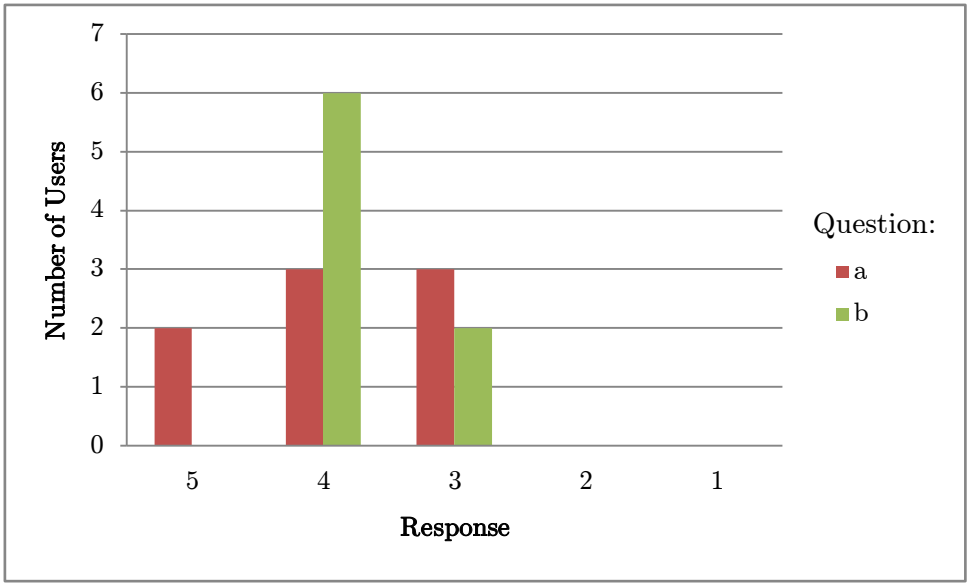

Figure 4.2: Graph showing the responses of users with XML experience when completing task 6

Post Task Questions

The post task questions were designed to get feedback on the users' experience of using the Curator Interface as a whole. This feedback is therefore the most valuable when determining the overall usability and comparability of the Curator Interface. Statistics regarding the distribution of the responses to each question was performed to establish a better understanding of the results. This distribution was then used in order to make inferences about how other users would find the usability

and comparability of the Curator Interface. The statistical analysis can be found in *Appendix A*. The statistical analysis shows the following:

- Question 1
    - The mean of the responses is 4.4, indicating that the majority of users found the overall usability of the Curator Interface to be positive.
    - The standard deviation of 0.5 shows that most of the total population[1] will respond in a range between 4.9 and 3.9. These are both positive results suggesting that most users will respond positively to the overall usability of the Curator Interface.
    - The 95% Confidence Interval (CI) of ±0.24 indicates that the total population mean is most likely to be in the range of 4.16 and 4.64, thus indicating that 4.4 is a good estimate. This provides validity to the results found in the statistical analysis.
- Question 2
    - The mean of the responses is 4.45, showing that users had a positive experience using the Curator Interface.
    - The standard deviation of 0.51 indicates that the majority of the total population will find the experience of using the Curator Interface positive, and respond between 4.96 and 3.94.
    - The 95% CI of only ±0.24 shows that 4.45 is a good estimate of the total population mean, highlighting the validity of the results.
- Question 3
    - The mean of the responses is 4.25, showing that most users found the look and feel of the Curator Interface to be satisfactory.
    - The standard deviation of 0.55 shows that the response to the look and feel was not as good as the previous two questions, but still positive overall.
    - The 95% CI of only ±0.26 shows that 4.25 is a good estimation of the total population mean, highlighting the validity of the results.
- Question 5
    - The mean of the responses is 4.6, showing that users had a very positive reaction to the performance of the Curator Interface.
    - The standard deviation of 0.5 reinforces this positive reaction by showing that the majority of the population will respond in the range of 4.1 and 5.
    - The 95% CI of only ±0.24 shows that 4.6 is a good estimation of the total population mean, highlighting the validity of the results.
- Question 6
    - The mean of the responses is 4.65, the highest of the post task questions. This shows that the Curator Interface responded predictably to the user's actions.
    - The standard deviation, while being fairly high at 0.59, still highlights the positive reaction from the users, as the majority of the total population would still provide a rating of above 4.
    - The 95% CI of only ±0.24 shows that 4.65 is a good estimation of the total population mean, highlighting the validity of the results.

[1] All potential users of the Curator Interface

- Question 8
    - The mean of the responses is 4.1 showing that although the general response was positive; it was the lowest of the post task questions.
    - The standard deviation of 0.64 shows that there was a wider range of responses to this question. This may have been due to the question which asks about comparability of the Curator Interface to other online file management systems. When asked to justify their response in the written question, the majority of the users stated that there was a difference in the look and feel, as well as the ability to search.
    - The 95% CI of only ±0.3 shows that 4.1 is a good estimation of the total population mean, highlighting the validity of the results.

Discussion

The overall usability of the Curator Interface was evaluated using the feedback from the tasks the users had to perform, as well as the answers to 1, 2, 3, 5 and 6 of the post task questions. This combination of the feedback and the post task questions was chosen as it contains a set of direct and indirect questions that assess the user experience while using the Curator Interface. From the results and previous analysis, it can be concluded that the experience provided by and usability of the Curator Interface is acceptable. Thus, using a simple file hierarchy did not have an effect on usability, except in the case of editing metadata. However, further development focussed around this area should provide a means to address this issue.

To assess the comparability of the Curator Interface, question 8 of the post task questions and its associated written justification was used. From the statistical analysis of question 8, it can be seen that the majority of the population would find the Curator Interface comparable to other online resource management tools. The responses gathered from the written justification to question 8 suggest that the only major difference between the Curator Interface and other similar tools was the look and feel, and the ability to search. Using the results obtained from the user evaluation, it can be concluded that the Curator Interface is comparable to other digital repository management systems. However, the accuracy of this result can be improved by evaluating users with experience in managing digital repositories as there was no prerequisite of prior experience to take part in the evaluation.

4.3 Performance Evaluation

Performance evaluation was done to determine the response time of the Curator Interface, as well as the amount of data being sent from the server to the client. These measures determine the Curator Interface's performance, as well as how responsive it will feel to the users. To establish a measure of the performance of the Curator Interface, the navigation feature was tested. Navigation was chosen as it is the most process intensive feature and therefore the choke point of the system. Furthermore, one of the key functions of the navigation feature is the establishment of relative paths to all the resources in the collections. Once these paths have been established, all actions that can be performed on these resources – uploading, downloading, deleting, editing, creating folders – are predominantly dependent on each user's internet connection.

The objective of the performance evaluation is to answer research question two:

- What is the impact of having a hierarchical file-based data store on the performance of the Curator Interface?

To answer this question, standard-case testing and extreme-case evaluation were performed.

4.3.1 General Information

All performance testing was done over a Local Area Network so that volatility of an Internet connection is accounted for. All results were obtained using the firebug[1] add-on for Mozilla Firefox, which provides detailed information regarding page load and server response times, along with the amount of data being transferred from the server. To ensure that the results obtained were standardised, each action was repeated 4 times and the average results were recorded. Furthermore, the browser cache and server were reset after each set of actions.

The server used in the performance testing had the following specifications:

- Intel Core 2 Duo CPU E7400 @ 2.80GHz
- 2 GB RAM
- Windows 7 Ultimate 32-bit
- Apache Tomcat 7.0.22
- Java Standard Edition Version 6 Update 25

4.3.2 Standard-Case Evaluation

Standard-case evaluation involves testing how the Curator Interface performs under normal conditions. This will give a good indication of how feasible it is for the Curator Interface to be used in current collections.

Aim

The aim of the standard-case evaluation is to determine the response time of the server, as well as the amount of data being sent from the server to the client using a well structured collection.

Method

- The assumptions made about a well structured collection were:
    - Storing up to 256 files in each folder is reasonable.
    - Folders containing a large number of files were unlikely to contain more folders.
- The structure of the starting file hierarchy being used in the standard-case evaluation is as follows:

[1] http://getfirebug.com/whatisfirebug

- Root Directory
  - Collection directory
    - Sub-collection directory
      - Book directory
        - 256 files
      - Book directory
        - 256 files
      - Book directory
        - 256 files
      - Book directory
        - 256 files

- The total number of files in the collection was increased exponentially as follows: 1024, 2048, 4096, 8192 and 16384. This was done to determine how well the Curator Interface could handle larger, structured collections.
- For the purposes of evaluation, it was assumed that the increase in files was handled in one of two ways. Either new books with 256 files each were added to the sub-collection (hereafter referred to as Case 1), or new sub-collections following the same format as the original were added with 1024 files in each(hereafter referred to as Case 2). Both of these methods were tested.
- The following measurements were done on the furthest children from the root directory:
  - The load time and data transfer when navigating into a directory (Initial Load).
  - The load time and data transfer that occurred as a result of paging through the contents of the directory (Page Load). The cache was not cleared before this task was performed as the initial load must take place before the page load can occur.

Results

Case 1: Figure 4.3 shows the times of the initial and paging loads, while Figure 4.4 shows the amount of data transferred on the initial and paging loads.

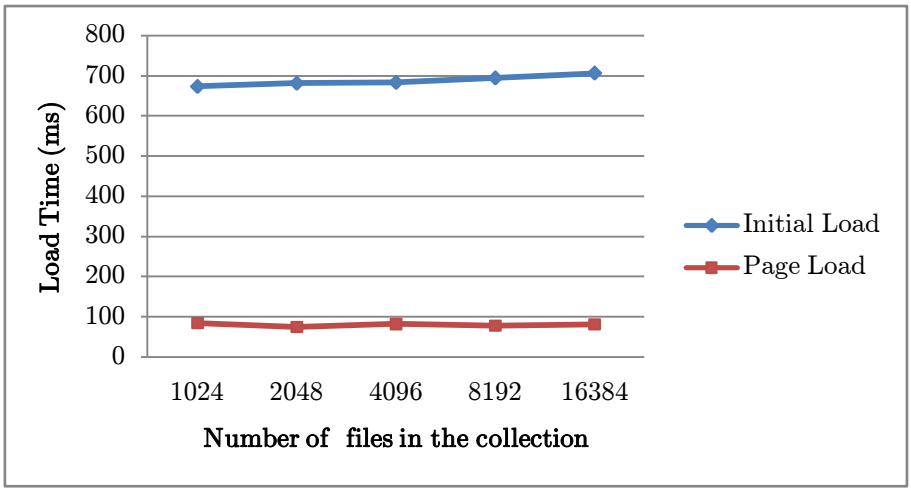

Figure 4.3: Graph showing the initial and paging loading times of directories in a collection for Case 1

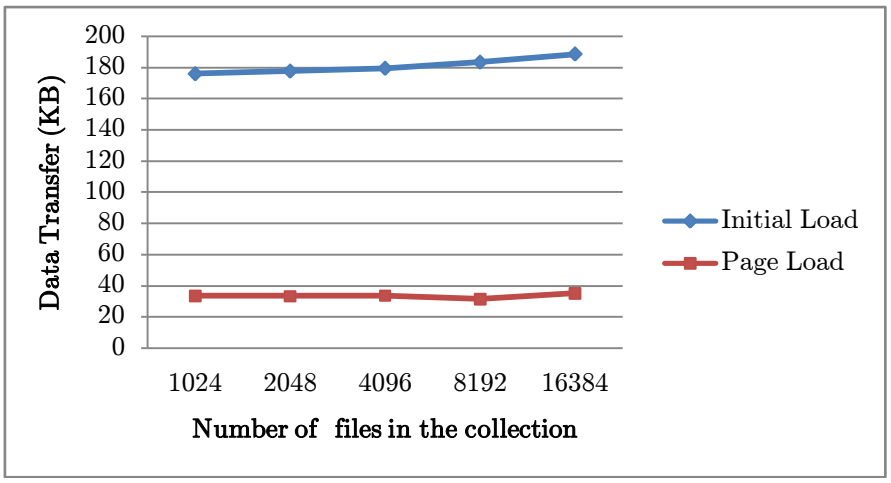

Figure 4.4: Graph showing the amount of data transferred on initial and paging loads in Case 1

Case 2: Figure 4.5 shows the times of the initial and paging loads, while Figure 4.6 shows the amount of data transferred on the initial and paging loads

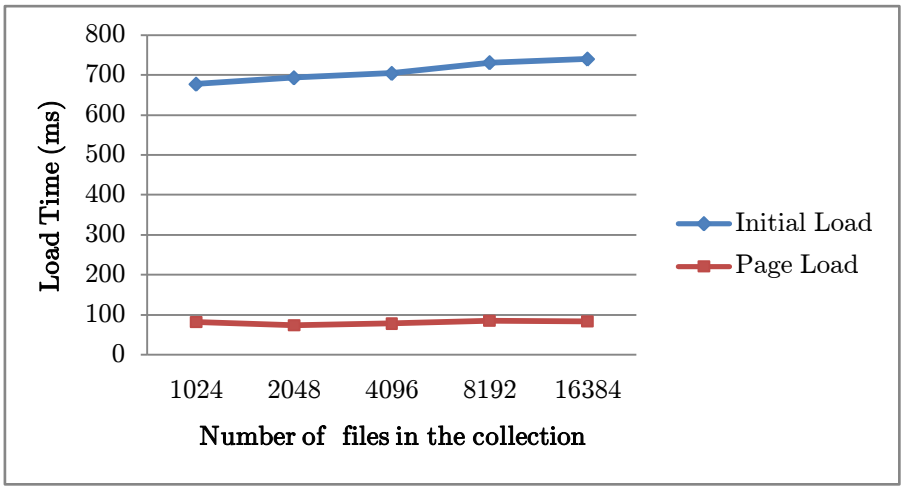

Figure 4.5: Graph showing the initial and paging loading times of directories in a collection for Case

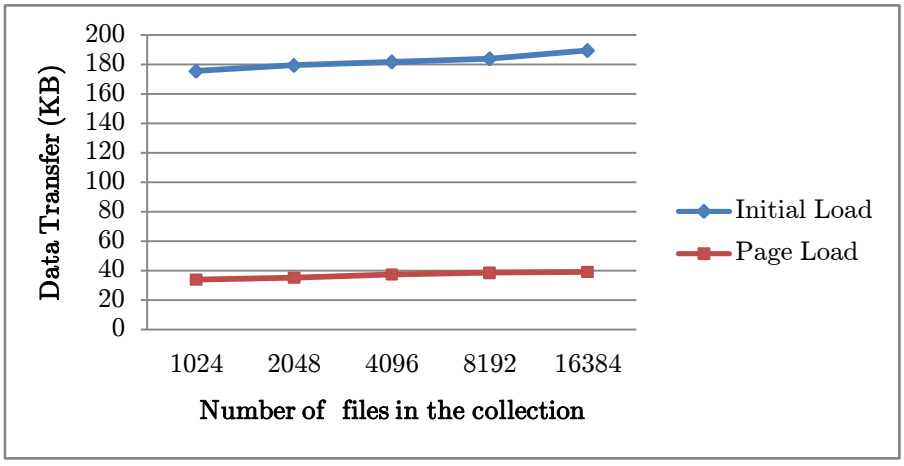

Figure 4.6: Graph showing the amount of data transferred on initial and paging loads in Case 2

Analysis

Figures 4.3 – 4.6 indicate that the load times and data transfers remained moderately constant with an increasing number of files in the collection. It was also shown that in both cases the Initial Load took less than 1 second, with less than 200KB being transferred. These results highlight the scalability of the Curator Interface, as it has been demonstrated that when using a well-structured file hierarchy the load times and the amount of data transferred remained similar for collections of between 1024 files and 16384 files. Furthermore, the small amount of data being transferred on the Initial Load highlights the applicability of the Curator Interface to an online environment where bandwidth may be limited.

The performance of the Curator Interface is further improved through the use of AJAX when paging through the contents of a particular directory (see *section 3.4.2* for more details). This is shown in Figures 4.3 – 4.6 as the Page Load times remain below 100ms and the amount of data transferred remains below 45KB with collections containing between 1024 and 16384 files.

4.3.3 Special-Case Evaluation

Special-case evaluation involves testing how the Curator Interface performs under unusual conditions. This will provide an indication of how flexible and robust the Curator Interface is. In addition, special-case evaluation may highlight some areas of weakness that are not present in the standard-case evaluation. Only one set of special-case testing could be conducted due to time constraints.

Aim

The aim of the special-case evaluation is to determine the response time of the server, as well as the amount of data being sent from the server to the client using an unstructured collection.

Method

- For the purposes of special-case testing, all the files in a collection were stored in a single folder. The files that were stored were all story metadata files, however, this does not make a difference as only the name and path of a file are returned from the server to the client.
- The amount of stories in that folder increased exponentially as follows: 1024, 2048, 4096, 8192 and 16384.
- The following measurements were then taken:
    - The load time and data transfer when navigating into a directory (Initial Load).
    - The load time and data transfer that occurred as a result of paging through the contents of the directory (Page Load). The cache was not cleared before this task was performed as the initial load must take place before the page load can occur.

Results and Analysis

Figures 4.7 shows the times for the Initial and Page Loads. It can be seen from the graph that the time taken for loading a directory increases in proportion to the number of files in that directory. Acceptable initial page loads 4.27 seconds, with further paging taking 2.03 seconds were achieved for directories containing 8192 files. However, the time taken for the Curator Interface to load directories with more than this amount of files may be considered unacceptable, as it would almost certainly be faster if databases were used. The bottleneck that is being experienced here is the time taken to iterate through the files in a directory as disk access is fairly slow.

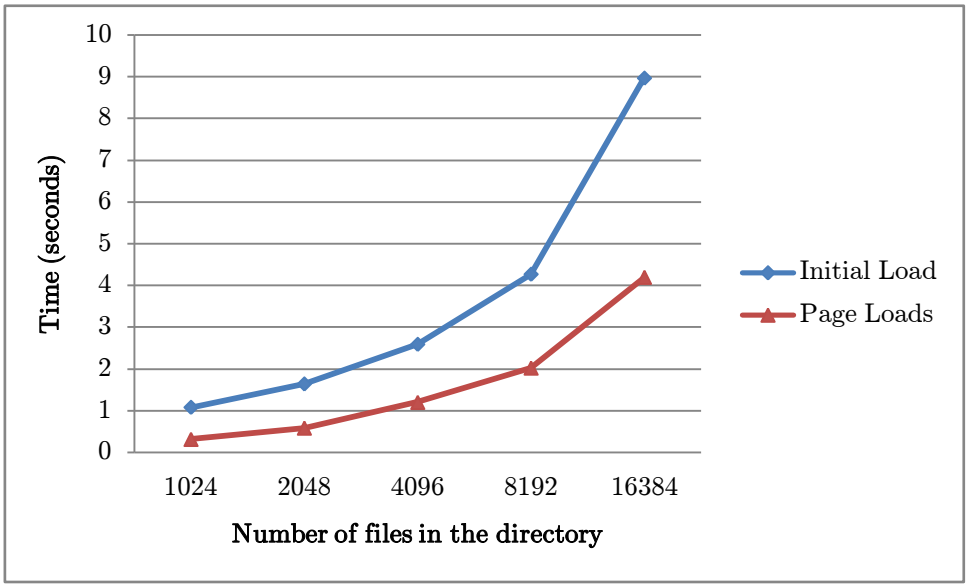

Figure 4.2: Graph showing the time taken for a Web page to load under unusual conditions

Figure 4.8 shows the amount of data being transferred from the server to the client. The graph shows that this amount remains fairly constant regardless of the number of files in the directory being viewed. This is due to the paging that occurs within each directory, restricting the number of results to 100 per page. It can also be seen that the amount of data being transferred on the initial load is significantly higher than the paging loads. The use of AJAX has been employed for paging to only refresh the right-hand side of the screen when a new page is selected. Thus all the static elements that were called on the initial load do not need to be obtained from the server. The extreme testing indicates that the amount of data being transferred from the client to the server will be acceptable regardless of the amount of files in the directory being viewed.

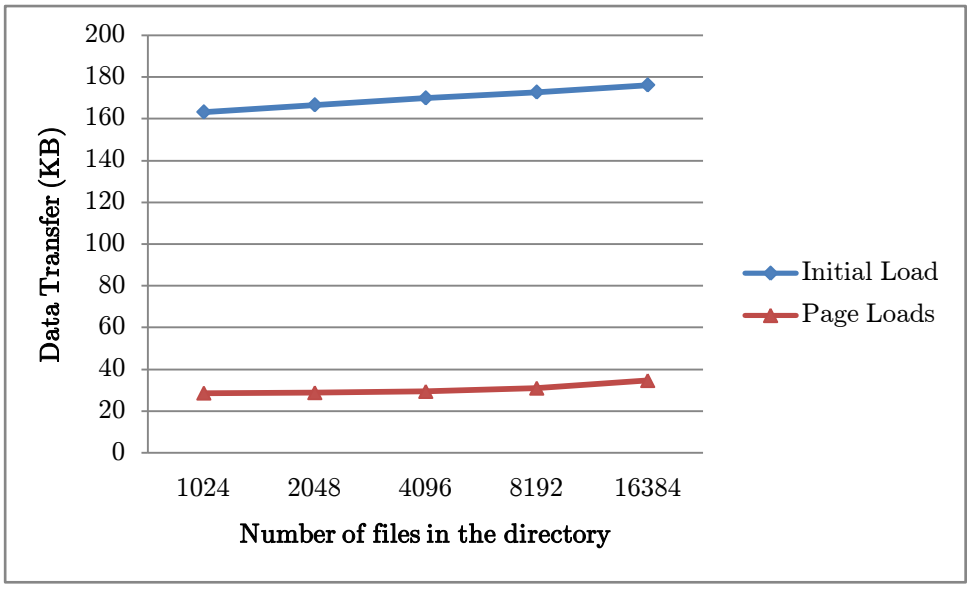

Figure 4.8: Graph showing the amount of data transferred from the server to the client under unusual conditions

4.3.4 Discussion

From the results obtained in the standard-case evaluations, it can be concluded that the performance of the Curator Interface is not compromised by using the simple file hierarchy as a data storage structure. Under standard situations, the Curator Interface demonstrates good scalability in terms of both speed and data transferral. However, this requires the contents of the collections to be well structured in directories.

The results of the special-case evaluation highlight the Curator Interface's weakness when dealing with large amounts of files in a single directory. While the amount of data being transferred remains constant and reasonably similar to the standard-case, the time taken to load the content of the directory increases directly with the number of files within that directory. The Curator Interface can process directories with up to 4096 files in under 3 seconds, and directories with up to 8192 files in just over 4 seconds. While these times are still acceptable for a web application they are not ideal. It is presumed that a database query would yield the same result more quickly. However, there is a low likelihood of a collection containing a directory with more than 2048 files, which the Curator Interface can process in under 2 seconds. From these assumptions, it can be concluded that the performance of the Curator Interface is not compromised by the use of a simple storage structure when the collection is well-structured. To deal with the concerns of unstructured collections, a pre-processor could be employed to index the contents of directories containing more than a threshold amount of files. This could potentially increase the performance of the Curator Interface significantly in special cases.

Chapter 5

Future Work

There is great potential for future work stemming from this project. As this is a tool that assists in allowing curators to better manage collections, there will always be room to improve the experience of the curator. Below, some of the possibilities for future work and development are discussed.

5.1 Communication Improvement

Improvements can be made in the communication between clients and server. These improvements would involve enhancing the algorithms and libraries used on the server so that it can respond more quickly and efficiently to client requests. This would lead to better performance in terms of both response time and data transferral.

5.2 Additional Features

Time constraints prevented implementation of all the tools found in the curator interfaces of existing digital repository systems. New features could be added to assist curators in managing collections by providing more tools and functionality. Some of the users who participated in the user evaluations indicated that they would like a search function, as well as a more efficient way of editing the metadata. The inclusion of these features, as well as others (such as workflow submission and integrity checks), may lead to a more user-friendly experience as well as a more effective digital repository management system.

5.3 Data Formats

The system is fairly limited in its capabilities of handling various data formats. Currently, it only has the capacity to display .jpg images and the metadata associated with them. The system does allow other data types to be stored online, however it does not allow users to view them. At this point, users would have to download the objects before they are able to view them. Further development of the system could allow users to view a much wider set of data types online.

5.4 File Structure

Although the system can display any file hierarchy to the user, it is most effective when the files are within a simplyCT framework. The system will, therefore, be most useful in cases such as the Lloyd and Bleek Collection. Future work could extend the framework to provide the same degree of control to other file hierarchies. This will provide the opportunity for the implementation of Bonolo in a wider number of collections.

5.5 Security

The nature of digital objects and their environment means that they can be easily copied, resulting in the existing paper-based copyrights being ineffective [Cleveland, 1998]. This may lead to certain issues if there are items stored in the collection that are of a sensitive nature, either due to their content or copyright agreement. It is therefore important that an access control feature is added so that curators can control which end users get access to which digital objects, and what actions those users can then perform on those objects.

5.6 Experienced User Evaluation

The users who participated in the user evaluation did not necessarily need to have any experience using existing digital repository systems. While the majority of the users indicated in the evaluation that they had some experience in managing an online collection of sorts, very few actually had some experience of using a digital repository system to manage items stored in an online collection. Performing additional user evaluations with experienced users would provide valuable feedback on how well the system performs compared to existing digital repository systems.

5.7 XML Editing

One of the issues noted in *section 3.7 Issues* was the lack of an easy-to-use XML editor. This was reinforced by the results of the user evaluation as the editing of stories was the task that proved to be the most difficult. While XML parsers can be constructed without much difficulty, a means of outputting the contents of the XML document to a Web page so that users can easily edit them is a more difficult task. An investigation into an online GUI-based XML editor that does not require the user to have a knowledge of XML documents would be beneficial to a number of projects.

5.8 Exporting to different formats

One of the suggestions raised in the focus group held during the design phase of iteration 2 was to allow curators to export the collections to different formats. This would be a useful function for improving the preservation of the objects within the collection as they could be exported into formats supported by other digital repository systems. The exporting feature would have to ensure that there is

no risk of losing some of the data in the exporting process. An essential component of this exporting function would be the ability to change the structure of the metadata so that it still adequately describes its associated digital object. Once again, the system would have to ensure that no metadata is lost through the exporting process. One possible method of ensuring that the objects and their metadata are exported correctly would be to construct and compare digital signatures for the digital objects before and after they have been exported. This method is used in other digital repository systems to aid in the preservation of digital objects [Jantz and Giarlo, 2005].

5.9 Pre-processing

The results of the special-case performance evaluation show that the performance of the Curator Interface – when reading directories with large numbers of files – can be improved. The inclusion of a pre-processor to index the contents of directories with a number of files over a certain threshold could significantly improve the time it takes for the Curator Interface to display the contents of these files. This would improve the performance and flexibility of the Curator Interface, as well as provide a more responsive system to the user.

Chapter 6

Conclusions

This project set out to investigate the possibility of building a functional online digital repository management system centred around a simple file hierarchy as a storage structure. Background research showed that digital repository systems had been successfully built around a simple file hierarchy. However, none of their management systems in these solutions provided sufficiently high flexibility while being online. The Bonolo Curator Interface – a framework that provides digital repository administrators to manage collections remotely – was designed to meet this need.

The system was designed iteratively allowing for feedback and improvement throughout the design process. This is an important aspect in designing a system that supports a high level of usability. It was realised early in the design process that, due to time constraints, not all the features and functions of traditional digital repository systems would be able to be implemented. To account for this, the core functionality of the system was identified and focussed on. These core functions have been successfully implemented and provide users with the tools to adequately manage collections online.

The possibilities for future work and development of the system have been identified and briefly examined.

To evaluate the system, both user and performance evaluations took place. The results of which were used to answer the research questions surrounding the system:

1. Will using a simple file hierarchy as a file store affect the overall usability of the Curator Interface?
2. What is the impact of having a hierarchical file-based data store on the performance of the Curator Interface?
3. Is the Bonolo Curator Interface comparable to those of other digital repository systems?

Using the results of the user evaluations it was concluded that the usability of the Curator Interface was not compromised through the use of a simple file hierarchy, as users only struggled in one task due to a limited knowledge of XML structure. Furthermore, it was shown from the user evaluation that the Curator Interface is comparable to other digital repository management systems. However, to obtain a more accurate result to research question three, users with experience in digital repository management systems should be used. From the performance evaluation, it was concluded that the performance of the Curator Interface was not compromised when using a well structured file hierarchy. However, the extreme-case testing showed that there is a performance impact if there are large amounts of files in a single directory as it would take 9 seconds to load a directory if it contained 16384 files. It was noted that the inclusion of pre-processing could significantly improve this performance.

Bibliography

Ackerman, M. S., & Fielding, R. T. (1995). Collection Maintenance in the Digital Library. *Proceedings of Digital Libraries Ô95*, (pp. 39-48). Austin, Texas, USA.

Apache Commons. (2011, October 13). *License*. Retrieved October 20, 2011, from Apache Commons: http://commons.apache.org/license.html

Borgman, C. L. (1999). What are digital libraries? Competing Visions. *Information Processing and Management, 35*, pages 227-243.

Cleveland, G. (1998, March). Digital Libraries: Definitions, Issues and Challenges. *UDT Occational Paper #8*.

Gartner, R. (2008). Metadata for digital libraries: state of the art and future directions. *JISC Technology and Standards Watch*.

Geisler, G., Giersch, S., McArthur, D., & McClelland, M. (2002). Creating Virtual Collections in Digital Libraries: Benefits and Implementation Issues. *Proc of the second ACMIEEECS joint conference on Digital libraries* (pp. 210-218). ACM Press.

GNU. (2011). *GNU General Public License*. Retrieved October 20, 2011, from GNU: http://www.gnu.org/copyleft/gpl.html

Herlin, S. J. (2003). *Scribd.* Retrieved October 17, 2011, from Ancient African Civilizations to ca. 1500: http://www.scribd.com/doc/4802778/ANCIENT-AFRICAN-CIVILIZATIONS-TO-ca-1500

Jantz, R., & Giarlo, M. J. (2005, June). Digital Preservation: Architecture and Technology for Trusted Digital Repositories. *D-Lib Magazine, 11*(6). Retrieved from http://www.dlib.org/dlib/june05/jantz/06jantz.html

Kahler, H., Kensing, F., & Muller, M. (2000). Methods & tools: constructive interaction and collaborative work: introducing a method for testing collaborative systems. *Interactions, 7*(3), 27-34.

Kuny, T., & Cleveland, G. (1998). The Digital Library: Myths and Challenges. *IFLA journal, 24*, 107–113.

Lagoze, C., & Van de Sompel, H. (2003). The making of the Open Archives Initiative Protocol for Metadata Harvesting. *21*(2), 118 - 128.

Loewald, T., & DeRidder, J. (2010, June 22). Metadata In, Library Out. A Simple, Robust Digital Library System. *code{4}lib*(10).

Payette, S., Blanchi, C., Lagoze, C., & Overly, E. A. (1999). Interoperability for Digital Objects and Repositories. *5*(5). Retrieved from http://www.dlib.org/dlib/may99/payette/05payette.html

Santhanagopalan, K., Fox, E. A., & McMillan, G. (2006). A Prototype for Preservation and Harvesting of . *9th International Symposium on Electronic Theses and Dissertations.* Quebec, Canada.

Skotness, P. (2011). *The Digital Bleek and Lloyd: Home*. Retrieved October 23, 2011, from The Digital Bleek and Lloyd: http://lloydbleekcollection.cs.uct.ac.za/

Smith, M., Barton, M., Bass, M., Branschofsky, M., McClellan, G., Stuve, D., Tansley R. and Walker, J.H. (2003, January). DSpace: An Open Source Dynamic Digital Repository. *D-Lib Magazine, 9*(1). Retrieved from http://www.dlib.org/dlib/january03/smith/01smith.html

SQLite. (2011). *SQLite Copyright*. Retrieved October 20, 2011, from SQLite: http://www.sqlite.org/copyright.html

Staples, T., Wayland, R., & Payette, S. (2003, April). The Fedora Project: An Open-source Digital Object Repository Management System. *D-Lib Magazine, 9*(4). Retrieved from http://www.dlib.org/dlib/april03/staples/04staples.html

Stearns, P. N. (1998). *Why Study History?* Retrieved October 23, 2011, from American Historical Association: http://www.historians.org/pubs/free/WhyStudyHistory.htm

Suleman, H. (2007). Digital Libraries Without Databases: The Bleek and Lloyd Collection. *In Kovacs, Laszlo, Norbert Fuhr and Carlo Meghini (eds): Proceedings of Research and Advanced Technology for Digital Libraries, 11th European Conference (ECDL 2007)*, (pp. 392-403). Budapest, Hungary.

Suleman, H. (2007). in-Browser Digital Library Services. *In Kovacs, Laszlo, Norbert Fuhr and Carlo Meghini (eds): Proceedings of Research and Advanced Technology for Digital Libraries, 11th European Conference (ECDL 2007)*, (pp. 462-465). Budapest, Hungary.

Suleman, H., Bowes, M., Hirst, M., & Suraj, S. (2010). Hybrid Online-Offline Collections. *Proceedings of Annual Conference of the South African Institute for Computer Scientists and Information Technologists (SAICSIT 2010).* Bela Bela, South Africa.

Tansley, R., Bass, M., Stuve, D., Branschofsky, M., Chudnov, D., McClellan, G., & Smith, M. (2003). The DSpace Institutional Digital Repository System: Current Functionality. *Proceedings of the 2003 Joint Conference on Digital Libraries (JCDL'03).* Houston, Texas, USA: IEEE.

The jQuery Project. (2011). *License*. Retrieved October 20, 2011, from The jQuery Project: http://jquery.org/license/

Waters, D. J. (1998). What are digital libraries? *CLIR (Council on Library and Information Resources) Issues, 4*.

Witten, I. H. (2004). Creating and customizing digital library collections with the Greenstone Librarian Interface. *Proceedings of International Symposium on Digital Libraries and Knowledge Communities in Networked Information Society* (pp. 97-104). Tsukuba, Ibaraki, Japan: University of Tsukuba.

Witten, I. H., McNab, R. J., Boddie, S. J., & David, B. (2000). Greenstone: A Comprehensive Open-Source Digital Library Software System. *Proceedings of the fifth ACM conference on Digital libraries* (pp. 113 - 121). San Antonio, Texas, USA: ACM.

Appendices

Appendix A

A Statistical Analysis of Post Task Questions

1) On a scale of 1-5 (1 being very negative, 5 being very positive) how would you rate the overall usability of Bonolo?

Variable	N	Mean	StdDev
RESPONSE	20	4.40	0.50

Variable	N	Median	Skewness	95% CI
RESPONSE	20	4.00	0.44	±0.24

2) On a scale of 1-5 (1 being very negative, 5 being very positive) how would you rate your user experience while using Bonolo? (I.e. did you feel comfortable while using the system?)

Variable	N	Mean	StdDev
RESPONSE	20	4.45	0.51

Variable	N	Median	Skewness	95% CI
RESPONSE	20	4.00	0.22	±0.24

3) On a scale of 1-5 (1 being very negative, 5 being very positive) how would you rate the look & feel of Bonolo? [feel free to browse around the site before answering]

Variable	N	Mean	StdDev
RESPONSE	20	4.25	0.55

Variable	N	Median	Skewness	95% CI
RESPONSE	20	4.00	0.13	±0.26

5) On a scale of 1-5 (1 being very negative, 5 being very positive) how would you rate the performance of Bonolo?

Variable	N	Mean	StdDev
RESPONSE	20	4.60	0.50

Variable	N	Median	Skewness	95% CI
RESPONSE	20	5.00	-0.44	±0.24

6) Indicate on a scale of 1-5 (1 being not at all, 5 being very highly) the degree to which Bonolo responded to your tasks in the manner you expected?

Variable	N	Mean	StdDev
RESPONSE	20	4.65	0.59

Variable	N	Median	Skewness	95% CI
RESPONSE	20	5.00	-1.52	±0.27

8) If you have, please indicate on a scale of 1-5 (1 being low, 5 being high) how comparable Bonolo was in terms of functionality, usability, user experience, performance, etc.

Variable	N	Mean	StdDev
RESPONSE	20	4.10	0.64

Variable	N	Median	Skewness	95% CI
RESPONSE	20	4.00	-0.08	±0.30

Appendix B

User Evaluation Handout

Bonolo Curator Interface User Evaluation

Dear User,

Please read and then sign the following consent form if you are willing to take part in the evaluation. In this evaluation, you will be asked to complete a set of tasks (below) and provide feedback on your experience while doing so. When providing feedback, please be as honest and accurate as possible, as this will provide the most valuable results. Please remember that the system is being evaluated and not you! If you struggled to complete a task, indicate that you had difficulty as it was probably a problem with the system. Furthermore, if you are having difficulty understanding a task, just ask me to clarify.

Consent Form

Introduction and Background

Thank you for taking the time to participate in this user evaluation. Your feedback is greatly appreciated, as it will be used to assess the system that has been designed. The system that has been designed is an honours project done under the University of Cape Town, Department of Computer Science.

The system that is to be evaluated is the **Bonolo Curator Interface**. This is a Web tool that allows you to manage collections of data that are stored on a Web server. The main objective of the evaluation is to determine the overall usability and usefulness of the system. This means that any feedback you have (positive or negative) will benefit the study being conducted.

Risks

There are no known risks of participating in this study; however, if a situation does arise you are free to leave the evaluation at any point.

Anonymity

Please rest assured that any feedback you provide will be kept anonymous. The purpose of this evaluation is to gain insight into the usability and usefulness of the **Bonolo Curator Interface**, not to judge the manner in which people used the system. If you have any concerns about the manner in which your feedback will be used, do not hesitate to email me: mrobinson@cs.uct.ac.za and I will give you a more detailed breakdown on how the results will be used.

Consent

User

I have read and understand each of the sections in the consent form. I agree to participate in the user evaluation of the **Bonolo Curator Interface**. I agree to let Miles Robinson use the results and feedback of this evaluation solely for the purpose of his honours project as long as they are kept anonymous.

Name:

Signature:

Date:

Evaluator

I agree to keep the results provided by this evaluation anonymous when writing up the report. The results of this evaluation will only be used for evaluation of the **Bonolo Curator Interface**.

Name:

Signature:

Date:

[*Please turn the page to begin the assessment*]

Task 1: Registration

Action: Register yourself and apply for access to the **bl_sample_archive** collection

Note: You do not need to use a valid email address for the purpose of this evaluation

How easily did you find the registration page? (Circle the most appropriate number)

(With extreme difficulty) **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very easily)

How easy was it to register your username?

(Very difficult**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very easy)

How easy was it to register for the collection?

(Very difficult**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very easy)

How did you find the overall registration process?

(Very ineffective**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective)

If you indicated that you found the registration process fairly or fairy ineffective (below a 3 in the last question), please explain why:

.....

.....

[Refresh the page to ensure you have been granted access to the **bl_sample_archive** collection]

Task 2: Navigation and viewing images

Action: Navigate to the **bl_sample_archive** collection. Browse through this folder to find the book (folder) **BC_151_A2_1_108**. Enter this book and view the file **A2_1_108_08885.JPG** Go back to the containing folder of **A2_1_108_08885.JPG.** Now navigate back to the **bl_sample_archive** collection without using the back arrow on your browser.

How do you feel about the way the navigation is presented?

(Very ineffective**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective)

How do you feel about the way the file and its information are displayed?

(Very ineffective**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective)

How did you find the overall experience of navigating through the collection?

(Very ineffective and frustrating**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective and intuitive)

If you indicated that you found the navigation of the site ineffective or frustrating (below a 3 in the last question), please explain why:

.....

.....

<u>Task 3</u>: Create and control access to a Collection

Action: Go to your account page. You want to start a new collection on the server. Start a new collection by creating folder **Test**.

How easy was it to create a File?

(Very difficult**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very easy)

How did you find the overall experience of starting a collection?

(Very ineffective**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective)

If you found the experience of creating a collection to be ineffective (lower than 3 in the question above) do you think that it will be easier when you next create a collection?

(Very ineffective**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective)

If you think you would still find creating a collection to be ineffective the second time around (lower than 3 in the question above) please explain why:

.....

.....

Action: Go to your account and see that **Test** is present in your collections, and that your recent activity shows that you have created a folder. Notice that a user is requesting the ability to curate collection **Test**, approve/decline his request.

How did you find managing who has the ability curate to your collections?

(Very ineffective**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective)

Task 4: Uploading

Action: Go to collection **Test**. Notice that there is currently nothing in the collection. Your next task is to upload the file **TestImage.JPG** to the collection. **TestImage.JPG** can be found on the desktop of the computer you are currently working on.

Action: Notice there is an icon indicating that the file you just uploaded does not have any metadata associated with it. Enter the file you have just uploaded. The message indicates there is no metadata associated with it. Upload **TestImage.JPG.metadata**to this file. **TestImage.JPG.metadata**can be found in the folder **metadata** on the desktop of the computer you are currently working on. Extra info: Metadata is information describing a certain object (i.e. data about data.)

Action: Go back to the collection folder (**Test**). Refresh and note the icon indicating a lack of metadata has disappeared. Your final upload task is to upload a batch offiles. A suitable batch file exists on the desktop of the computer you are currently working on. Upload this file to the collection. Browse through this recently uploaded set of files and folders on the Website.

How easily did you find out how to upload a single file?

(Very difficult**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very easy)

How did you feel about how you uploaded the metadata associated file?

(Very inefficient**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very efficient)

How did you feel about how you uploaded a batch of files?

(Very ineffective**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective)

The system reacted as expected when I uploaded a batch of files. Indicate how you agree with this statement:

(Strongly disagree**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Strongly agree)

What is your overall impression of uploading files?

(Very inefficient**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very efficient)

If you had a negative impression of uploading files (below a 3 in the last question), please elaborate as to why:

.....

.....

Task 5: Downloading

Action: Go to the collections folder (collections/) and download the entire **Test** collection. Go through the downloaded file to ensure that all the files that were in the collection on the Website have been downloaded.

How easily did you find out how to download a file/folder?

(With significant difficulty**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very easily)

The system reacted as expected when I downloaded a collection. Indicate how you agree with this statement:

(Strongly disagree**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Strongly agree)

What is your overall impression of downloading files?

(Very ineffective**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective)

Task 6: Stories and Editing

Description: Stories are files that describe a set of images in the collection.

Action: Navigate to the **stories** folder in the **bl_sample_archive**. Open story 1000. Upon scanning the information about the story you notice that some of the information is incorrect. Make the following changes to the story: Add your name to the list of authors; add the category TestCat; add page RandomImage.JPG and book TestBook.

How easily did you find out how to edit a story?

(Very difficult**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very easy)

How did you find the experience of editing a story?

(Very ineffective**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very effective)

Have you ever worked with XML documents before?

Yes | No

If you found the editing a story to be ineffective (below a 3 in the last question), suggest how you would improve it:

.....

.....

Task 7: Deleting

Action: Delete the **Test** Collection. Go to your account to confirm that you no longer have access to the **Test** Collection (as it would have been deleted).

The system reacted as expected when I deleted the collection. Indicate how you agree with this statement:

(Strongly disagree**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Strongly agree)

What is your overall impression of deleting?

(Very inefficient**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Very efficient)

Task 8: Flexibility

Action: Your final task is to test the flexibility of Bonolo. Using the resources in the folder **resources** on your desktop construct a collection (i.e. create a root folder, with subfolders and put images into the folders wherever you feel like it.) Zip this collection and upload it to the collections folder on Bonolo. Browse through and perform any actions you wish on the collection.

The system showed a high degree of flexibility with regard to the structure of a collection. Indicate how you agree with this statement:

(Strongly disagree**)** **1 ------ 2 ------ 3 ------ 4 ------ 5** (Strongly agree)

Post Task Questions – Write answers down next to the questions

1. On a scale of 1-5 (1 being very negative, 5 being very positive) how would you rate the overall usability of Bonolo?
2. On a scale of 1-5 (1 being very negative, 5 being very positive) how would you rate your user experience while using Bonolo? (i.e. did you feel comfortable while using the system?)
3. On a scale of 1-5 (1 being very negative, 5 being very positive) how would you rate the look & feel of Bonolo? [feel free to browse around the site before answering]
4. Are there any changes you would make with regard to how users interact with Bonolo?

.....

.....

.....

5. On a scale of 1-5 (1 being very negative, 5 being very positive) how would you rate the performance of Bonolo?
6. Indicate on a scale of 1-5 (1 being not at all, 5 being very highly) the degree to which Bonolo responded to your tasks in the manner you expected?
7. Have you ever used a Website allowing you to manage data stored on a server? Examples of this might be: managing photos on Facebook, managing resources on Vula, Dropbox, editing a wiki, using a digital repository system (like D-space, Fedora, Greenstone, etc.)

   Yes | No

8. If you have, please indicate on a scale of 1-5 (1 being low, 5 being high) how comparable Bonolo was in terms of functionality, usability, user experience, performance, etc.

   Please explain your answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thank you for your time!

Your contribution to my honours project is greatly appreciated.