LARGE IMAGE SUPPORT IN DIGITAL REPOSITORIES

By

Marius Nel

Supervised by

Hussein Suleman

DISSERTATION SUBMITTED FOR THE PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF PHILOSOPHY (IN INFORMATION TECHNOLOGY)
IN THE DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF CAPE TOWN
January 2010

To Bronwyn, Ché-Ché
and my parents

Abstract

Many universities, libraries, government organisations and companies are implementing digital repositories to collect, preserve, administer and distribute their collections via the World Wide Web. In the process of building these digital archives and collections, images such as maps are often are captured in an uncompressed, high-resolution format to preserve as much detail as possible. This process, of high-resolution archiving gives rise to the problem of providing the end-user with access to these large (high-resolution) images, such as maps.

This dissertation investigates methods of storing and delivering large images over the Internet while limiting the amount of data being transferred; and also documents efforts to incorporate large image support within the *DSpace* platform.

An end-user usability study of various large image support solutions was conducted to establish how current digital repository large image solutions compared to commercial large image solutions. The study showed that the commercial large image solutions were superior to current digital repository solutions.

A prototype large image solution was developed with a specific aim to provide *DSpace* with mechanisms to import and deliver large images in a bandwidth-conscious manner. It was found that by implementing and extending currently available open source large image processing software, large image support could be provided to the *DSpace* platform with minimal or no modification to the *DSpace* source code.

An end user evaluation study was conducted to establish the usability and effectiveness of the prototype large image support solution. It was found that the prototype system provided an easy to use solution that provides *DSpace* with an effective large image archiving and delivery mechanism.

Acknowledgements

I would like to express my heartfelt thanks to:

My supervisor, Hussein

My wife, Bronwyn and my daughter Ché

Gerrie, who talked me into this

The Quirkstars who were kind enough to partake in the evaluations

Table of Contents

List of Figures

List of Tables

Introduction

Increasingly, libraries, higher learning institutions and cultural heritage institutions have taken action to design and implement digital repositories to provide long-term storage, preservation and permanent access to digital materials. In many cases this action was prompted by projects that focused on the digitizing of materials to build archives and collections that can be accessed via the World Wide Web (Verheul, 2006). Multimedia, especially images and video, are important parts of this on-line digital information while advances in image acquisition and storage technology have led to tremendous growth in very large and detailed image databases (Vassiliadis *et al*, 2005).

In the process of building these digital archives and collections, images often are captured in an uncompressed or lossless format to preserve as much detail as possible. This process, of lossless archiving, results in high-resolution images stored in large files, which in turn gives rise to the problem of providing the end-user with access to these high-resolution images.

1.1. Motivation

A repository should be able to provide a user with the original lossless versions of digitised material (Smith, 2008), for example, by providing access to and presenting large, uncompressed images to the users of the digital repository.

The growing availability of high-bandwidth networks significantly eases access to images consisting of hundreds of kilobytes of data or megabytes of data. Images consisting of tens of megabytes, hundreds of megabytes and gigabytes, however, often exceed the capacity of high volume transmission technologies (*Zoomify Inc.,* 2003).

Advanced compression techniques achieve 4 to 1 or even greater *lossless* compression and 10 to 1 or even 100 to 1 *lossy* compression. A gigabyte image file compressed with wavelet technology would still be tens of megabytes in size.

Web-based applications that aim to provide the ability to access very large uncompressed images have to overcome two real obstacles. The first obstacle is the large file size of these images and how to transfer that amount of data efficiently to the client. The second obstacle is the screen size or viewing window of the client browser (see Figure 1).

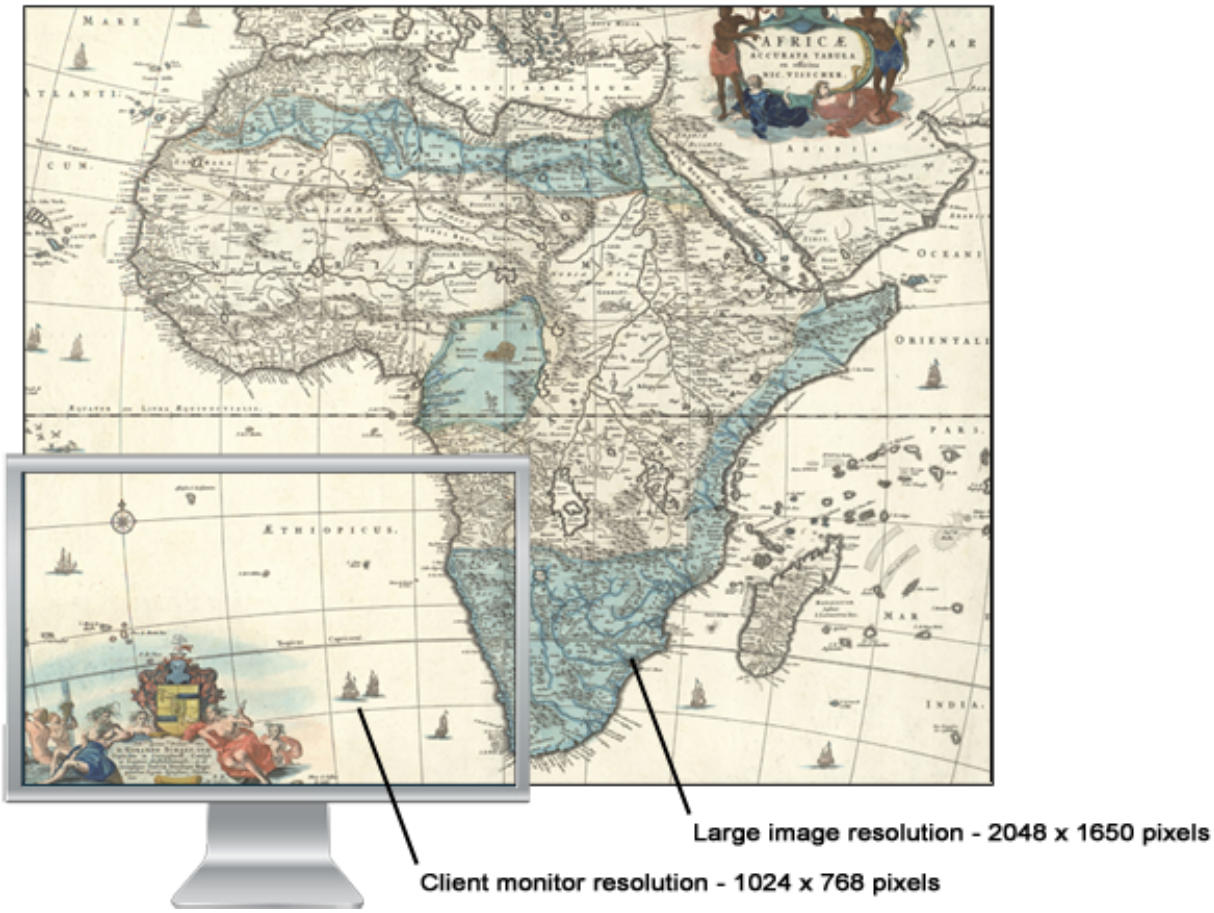

Figure 1- Client monitor size and large image resolution obstacle

There are, however, many commercial and open-source solutions available that effectively overcome these obstacles, but few Digital (DR) tools and operational repositories with large image support.

The goal of this project, therefore, is to investigate the feasibility of incorporating an effective large image solution within a popular DR platform.

1.2. Objectives

The aims of this research are threefold:

1. To establish the efficacy of current large image support solutions that have been implemented within digital repositories (DRs) and also commercial solutions that exist beyond the scope of digital repositories.
2. To design and implement a large image support solution within a popular DR platform and evaluate its effectiveness.
3. To determine the usability and efficacy of the designed system through prototypes, evaluations and empirical studies.

1.3. Organisation of this dissertation

Chapter 2 summarises the relevant background literature and previous and related work used in formulating the research. The methodology adopted for this research is explained in Chapter 3. Chapter 4 presents the details and results of a survey, which documents users' ratings of the usability and effectiveness of various large image solutions within DRs and commercial solutions. Chapter 5 documents the design of a system that provides the DSpace platform with support for large images. Chapter 6 provides results of user evaluation studies of the system. Finally, chapter 7 summarises the main conclusions and discusses possible future directions.

2. Background

This chapter reviews and evaluates existing literature regarding the topic of large image support within Digital Repositories. The first section investigates research with a focus on the role and function of DRs. The second section discusses various DR platforms. The third section investigates efforts of modifying the DSpace platform. The fourth section reviews different image file formats in terms of digital preservation strategies. The fifth section focuses on literature that deals with the concepts and technologies that form the basis of large image support solutions and also literature that documents solutions in related commercial and non-commercial Internet applications. The final section presents a summary of the literature review.

2.1. Role and function of Digital Repositories

The advent of digital technology and high-speed networks in the late 20th century resulted in an enormous increase in information production that is still continuing today (Lyman and Varian, 2003). To deal with this rapid increase in information, organizations all over the world started implementing Records Management Systems and later Content or Knowledge Management Systems (Barry, 2002). According to Aschenbrenner and Kaiser (2005), Institutional or Digital Repositories is another product of this movement. The concept of an Institutional Repository was first established in the higher education and research environment where institutions started developing repositories to capture, manage and disseminate the organisation's intellectual output (Aschenbrenner and Kaiser, 2005).

Tedd and Large (2005) inform that although the early digital library research projects were based in university settings, since the late 1990s, there has been tremendous growth in the number of digital libraries with many different roles and purposes. Culture-based organizations, such as museums, are developing digital collections of their assets which can then be freely accessed by individuals and educational institutions from anywhere in the world. The development of digital libraries has also enabled national museums to provide access to important collections of archives, printed materials and photographs related to a particular country (Tedd and Large, 2005).

In the research sector, The ARL report (2009) on *The Research Library's Role in Digital Repository Services* states that DRs are developing rapidly as a key element of research cyber-infrastructure. The report points out that DRs are currently still developing and adaptations, changes in directions and corrections characterize the experience of DR developers as they try to overcome the challenges of building services around new content and old content in new forms as institutions produce ever-increasing volumes of data, images, multimedia works, learning objects, and digital records while mass digitization has introduced a new scale of digital content collecting. Another factor impacting the development of DRs is the shift of focus from building repositories to the delivery of services curtailed the needs of the end-users.

Aschenbrenner and Kaiser (2005) state that in the late 1990s the early stages of DR development was characterized by a considerable input of software development work by skilled computer scientists that only a few large institutions could afford. A few years later, some organizations started developing general-purpose repositories in customizable software packages. Many organizations around the world have increasingly opted to implement these software packages to establish their own repository services. The following section investigates some of these DR software packages.

2.2. Digital Repository platforms

DR platforms consist of hardware, software and open standards and the more commonly adopted software solutions fall into two broad groups: open source and commercial software. Popular open source software includes EPrints, Fedora and DSpace (Peneva *et al.,* 2009). This section will briefly investigate these three software packages.

2.2.1. EPrints

Saleh *et al.* (2005) describe EPrints as an open source software platform developed at the University of Southampton, UK, and is intended for use by universities and research insitutions as a digital repository system for educational material. EPrints enables registered authors to *self archive* (deposit a digital document), which involves submission via a simple Web interface where the depositer inserts the *metadata* (date, author-name, title, journal-name, etc.) and then attaches the full-text document. The submitted document is indexed for searching and positioned within a subject hierarchy defined in the system. EPrints makes the metadata available for harvesting by the OAI-PMH interface. The Open Archives Initiative Protocol for Metadata Harvesting provides an application-independent interoperability framework based on metadata harvesting. There are two classes of participants in the OAI-PMH framework: Data Providers are repositories that expose structured metadata via OAI-PMH; and Service Providers then use metadata harvested via the OAI-PMH as a basis for building value-added services. (Lagoze *et al.*, 2002).

As an example of exploiting the OAI-PMH framework, the EPrints UK project was built using the EPrints software to develop a series of national, discipline-focused services through which the higher and further education community can access the collective output of eprint papers available from OAI-compliant repositories (Martin, 2003) .

2.2.2. Fedora Project

Lagoze *et al*. (2005) describe the Fedora Project as an ongoing research and development endeavor to build a framework for creation, management and preservation of existing and evolving forms of digital content. The Fedora project evolved from the DARPA-funded research in the early 1990s where the notion of a *digital object,* was first defined. A digital object is a data structure whose principal components are digital material, or data, plus a unique identifier

for this material, called a handle (Kahn and Wilensky 1995). The first prototype consisted of a CORBA-based Fedora (Flexible Extensible Digital Object Repository Architecture) system (Payette and Lagoze, 1998).

Fedora further developed from a research project to a production repository system when the University of Virginia Library experimented with the Fedora architecture in an effort to find a solution for managing its increasingly complex digital content (Staples and Wayland, 2000).

Fedora is implemented as a set of web services that provide full programmatic management of digital objects as well as search and access to multiple representations of objects. Fedora's APIs are described using the Web Service Description Language (WSDL) which makes it well suited to exist in a broader web service framework and act as a foundation layer for a variety of multi-tiered systems, service-oriented architectures, and end-user applications. This distinguishes Fedora from other complex object-based platforms such as DSpace and EPrints that provide ready-made applications for storing and manipulating complex objects through a fixed user interface (Lagoze *et al*., 2005).

Currently, the Fedora Repository Project and the Fedora Commons community together with the DSpace project are under the supervision of the not-for-profit organization DuraSpace.

2.2.3. DSpace

Tansley *et al.* (2005), describe DSpace as an Open Archival Information System, developed by Hewlett-Packard Laboratories and MIT Libraries in a two-year collaboration project to develop an open-source repository system for the capture and preservation of digital materials. DSpace main aim was to address the need of research organizations and educational institutions that were increasingly producing digital-born output such as audio, video, statistical databases, software and the more traditional document-based material but didn't have a suitable place to place all these items.

To accommodate these varied types of data and media that are stored within its database, DSpace includes support for a wide range of file-formats. Capture happens primarily through a Web interface, which collects the metadata elements and manages file uploads; as a secondary means of capturing items, there also is a bulk upload feature.

DSpace was designed as a production quality system that offered all the functionality required for a long-term institutional repository in a relatively simple fashion. DSpace went into production at MIT Libraries in November 2002, and has been available as open source software since that time.

To alleviate the need for a single comprehensive or authoritative list that provides a record of the range of open access repositories, the Centre for Research Communications at the University of Nottingham created OpenDOAR as an authoritative directory of academic open access

repositories. According to the latest (January 15, 2011) report, DSpace is the most popular digital repository platform, accounting for more than a third of installed Repository platforms (see Figure 2).

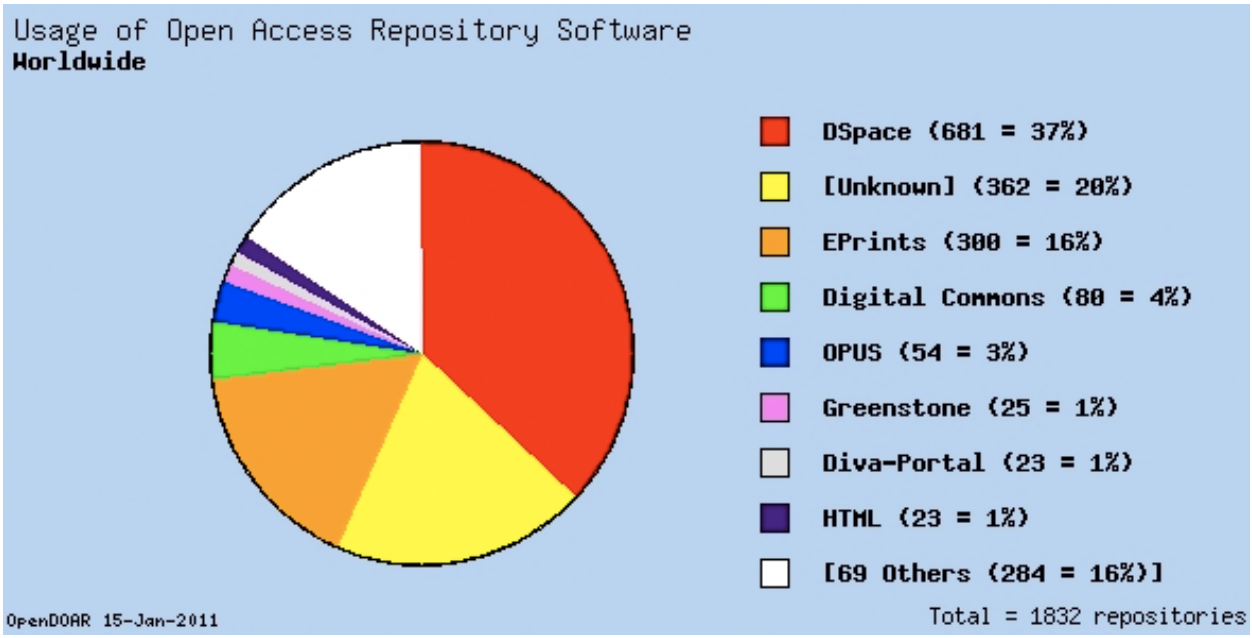

Figure 2 - OpenDOAR data: Usage of Open Access Repository software, worldwide

The next section investigates research that documents approaches to modify the DSpace platform to provide additional functionality to this popular open source DR platform.

2.3. Modifying the DSpace platform

Richard Jones (2004) from Edinburgh University Library explains how the *Theses Alive Plugin for Institutional Repositories (Tapir)* has provided E-Theses functionality for DSpace, and discusses the development methods and process for writing such third-party tools. He briefly explains how the DSpace source code is maintained on a version-controlled repository on a publicly accessible server, hosted by SourceForge, where individuals with administrative control over the versioning system, known as committers, can affect any changes to the core code and approves code submissions from contributors. Periodically, a version of the source code will be declared as stable and packaged as an official release, which can be downloaded and used by people not interested in working with the development copy. Jones also points out that a decision needs to be made regarding the development model that will be used for modifying the *DSpace* software. Will you write patches to existing source code and commit the changes to the

versioning system; or will you write and maintain your own software pack that can be installed onto *DSpace*? Edinburgh University Library chose the second approach because it was not anticipated that their work would be of interest to the whole *DSpace* community and that their work and development was going to move at the same speed as *DSpace* development.

Phillips *et al.* (2007) introduce Manakin, a modular interface layer to the Dspace platform that separates the user interface from the service implementation, and enables easy modification of the look-and-feel of the repository through the use of themes. Themes are self-contained packages and multiple themes may be installed to be available for use in different parts of the repository. Themes can be applied either to the entire repository or to specific communities, collections or items within the repository. The theme application rules cascade downward so, when a theme is applied to a community, all collections and items contained within the community also inherit the theme's look-and-feel. These collections and items can either use the inherited theme or provide one of their own.

Manakin therefore provides an alternative method to customise the standard DSpace user interface, which is based upon JSP technology, and is difficult and expensive to modify. Manakin has been included in the official DSpace release since version 1.5.

Allinson *et al.* (2008) describe the SWORD (Simple Web service Offering Repository Deposit) Project, which has produced a lightweight protocol for repository deposit and implementations of a deposit interface into DSpace, amongst others. The SWORD implementation for DSpace was developed as an 'add-on' to DSpace which makes use of the new modular build system introduced in DSpace version 1.5. This allows the SWORD module to be installed with ease alongside a current DSpace installation.

The following section investigates different digital image formats in terms of digital preservation strategies.

2.4. Image file formats for digital preservation

Digital preservation strategies dealing with high resolution images often lead to a choice between keeping terabytes of images in their original TIFF format or to compress them to save storage costs (Buonora and Liberati, 2008). Compression could lead to a loss of visual information and therefore waste of money expended in the creation process of the digital assets. On the other hand, compression can lead to cost savings in terms of storage. This section investigates various image file formats and their suitability in the realm of digital preservation. Preservation is defined as those activities revolving around the maintenance of materials in a usable form for an extended period of time (Hodge and Anderson, 2007). A suitable preservation format therefore, is a format that provides the best options to achieve preservation, including the ability to capture the material in the archive and then render and disseminate it now and in the future (Hodge and Anderson, 2007).

Zierau and Jensen (2010), argues that dissemination should be taken into account when evaluating digital preservation options in a library context because the library's digital material must be disseminated to the public or researchers through fast access. Preservation and dissemination respectively place different demands on the formats in which digital materials are preserved and presented. While JPEG 2000 or TIFF have been used by many libraries as the chosen preservation format for books and images, dissemination may benefit from formats, for example JPEG, PNG or GIF, that consume less storage and is better supported in browser software.

2.4.1. JPEG 2000

Robert Buckley (2008) presents JPEG 2000 as a practical preservation standard. He highlights the JPEG 2000 feature set, which include: A single architecture for lossless and visually lossless image compression, progressive display, multi-resolution imaging and scalable image quality, the ability to handle large and high-dynamic range images and generous metadata support. He argues that the single most important feature of JPEG 2000 is "smart decoding". Smart decoding provides multiple decompression options from a single JPEG 2000 code-stream and enable an application to access and decode only as much of the code-stream as is needed for the task at hand. Smart decoding therefore makes it possible for a single JPEG 2000 compressed image to supply multiple derivatives of the original image. It can, for example provide multiple reduced resolution versions of the original, or a high-resolution, high quality view of a portion of the image with a region of interest. This feature, to support multiple derivatives, has important consequences for online image collections. Typically, applications that offer access to online collections provide multiple, different-sized image views, each supported by a pre-generated image file. This adds increased costs because multiple files needs be stored, maintained and synchronized. JPEG eliminates these extra costs by supplying multiple resolution images within a single JPEG 2000 code-stream.

Buckley also points out that besides a wide array of output options, JPEG 2000 also supports a variety of input images. It is equipped to handle large images with hundreds of megabytes and one vendor claims that can support terabyte images. JPEG 2000 also supports high-bit depth images, which is useful as colour encodings with 12 and 16 bits per component become more commonplace. This suggests that JPEG 2000 could meet longer-term needs for image archiving because of its ability to encode a wide range of images.

Lowe and Bennett's (2008) JPEG 2000 status report exposed several key areas that needed to be addressed to further enhance adoption of the JPEG 2000 standard. The report reveals that existing implementers had concerns about codec inconsistencies among software vendors and how these inconsistencies impacted on future migration toolkits. There were also concerns about the general lack of adoption of the standard. Non-implementers were concerned about the format's stability and permanence. This perception is being fueled by the lack of available

software functionality, from initial capture and manipulation, to final delivery to online users.

Lowe and Bennet further note that frustration was also expressed on the lack of browser support for the standard. Much of this frustration was focused on the additional server-side processing that was required to take advantage of standard's flexible, zoom and panning capabilities from a single JPEG 2000 file. Native JPEG 2000 support is currently absent from Internet Explorer, Mozilla Firefox, Safari and Chrome. The QuickTime plug-in for each can render JPEG 2000, though only at one zoom level.

Lowe and Bennett believes that the lack of browser support is due to the design of the wavelet compression of JPEG 2000 which requires more computing from the viewer's device and software. The major browser developers try to keep their code base light as a competitive advantage, which has been a factor that has hampered adoption of the standard due to the additional code required to take advantage of JPEG 2000's features.

Mindful of the advantages that JPEG 2000 offers, the Bavarian State Library (Bayerische Staatsbibliothek, BSB) considered the option of migrating from TIFF to JPEG 2000 as the archive format for digitized images of rare books. Hannes Kulovits and Andreas Rauber from the Bavarian State Library, together with Anna Kugler *et al*. from Vienna University of Technology (2009), created a preservation plan for a representative collection of digitized 16th century printings with the goal to evaluate possible strategies for migration from TIFF to JPEG 2000 using lossless compression, including the alternative of keeping the status quo (not migrating from TIFF).

Following the digitization policies of BSB, four open source tools that are able to perform TIFF to JPEG 2000 migration using lossless compression were considered. Their evaluation revealed that not migrating the TIFF images to JPEG 2000 was the best option for BSB because the migration tools did not meet the BSB's preservation plan quality requirements.

2.4.2. TIFF

TIFF (Tagged Image File Format) is one of the earliest formats used to preserve digital images (Hodge and Anderson, 2007). It was originally developed and trademarked by the Aldus Corporation, which subsequently merged with Adobe systems. The original purpose was to create a file format to store raster images originating from scanners and image editing software and to provide an environment within which applications can exchange image data. High priority was also afforded to the structure of the file so that developers can add future enhancements easily. This feature allows for great flexibility and has been used extensively although not all extensions are used by all image editing and viewing software (Gillesse *et al.,* 2001).

While most image file-formats are designed to support a single compression technique, TIFF allows for multiple compression methods including JPEG compression for *lossy* compression

and LZW compression for *lossless* compression (Borghoff, 2006). The LZW compression dates back to 1984 and is essentially an improved version of the LZ78 algorithm from 1978. The name, LZW is derived from the names of the algorithm's developers, Jacob Ziv, Abraham Zempel and Terry Welch. It was developed as a lossless data (not only images) compression algorithm (Gillesse *et al.,* 2001).

Gillesse *et al.* (2001) describe the structure of the TIFF file, which begins with an image file header (IFH) that refers to the image file directory (IFD) with the associated bitmap. The IFD contains information about the image and pointers to the actual image. A TIFF file can contain multiple IFDs to describe a multi-page TIFF file, which for example would allow multiple images to be stored within a single file.

The file header and IFD contain sections of data, called 'tags' which contain basic geometric information, the image data's organization and whether a compression technique is used, for example. There is also the option for users to use their own, *private tags,* to contain additional metadata (Gillesse *et al.,* 2001).

The Federal Agencies Digitisation Initiative (FADGI) Still Image Working Group (2010) recommends TIFF as the preferred format for the production of an archival master file based on a list of technical considerations that include the fact that TIFF is the "De Facto" raster image format used for archival master files, is widely supported by image editing and viewer software, includes comprehensive metadata possibilities, it supports uncompressed lossless compression, a large number of colour spaces and profiles is accommodated, large image files is accommodated and high-bit images is supported.

Gillesse *et al.* (2001) note that the most important disadvantages of TIFF are the large file size that results from the use of lossless compression and the lack of browser software support.

2.4.3. JPEG

JPEG was developed in the early 1990s specifically for the storage and transmission of photographic images. The name JPEG (Joint Photographic Experts Group) represents the committee that created the standard for compression of continuous tone greyscale and colour images. The JPEG standard specifies the codec with which the images are compressed/encoded in a data-stream and also defines the data-stream's file format (Gillesse *et al.,* 2001).

Wiggins *et al.* (2001) note that the strength of the JPEG format is its capacity to considerably compress larger image files. Image compression to as little as 20:1 of the original file size can be achieved. A further advantage is its comprehensive support by image editing-, viewing- and browser software (Gillesse *et al.,* 2001). The major drawback of JPEG is that loss of image data, which translates into image degradation, that results from the *lossy* compression technique. A further weakness is the likelihood of image distortion created by the compression technique,

which affects image files composed of only a few colors or those with large areas of the same color, such as images of text documents (Wiggins *et al*., 2001). JPEG however effectively supports full-colour 24-bit images, and works well for photographic images, where continuous shades of colours are present, making it the common standard for compressing photographic images on the Web today (Freeman, 2008).

2.4.4. PNG

Gillesse *et al.* (2001) describe PNG (Portable Network Graphics) as a data-stream and an associated file format for a lossless compressed, portable, individual raster image. Initially developed for transmission via the Internet, the PNG development group developed the format under the supervision of the World Wide Web Consortium (W3C) as an alternative for the then-patented GIF format and its associated LZW compression. PNG compression is lossless and therefore does not cause degradation of the image quality during compression.

Wiggins *et al.* (2001) describes the 'chunk architecture' of the PNG file where the PNG data-stream is made up of a PNG signature that indicate that it is a PNG data-stream, followed by a sequence of chunks (segments) of data that encode the image information to the decoder program (e.g. a Web browser). These chunks of data can contain keywords and image metadata that are interpreted by certain decoder programs while ignored by others. Internet search engines can detect this metadata and therefore the image can be found faster.

PNG's support for image metadata and its lossless compression method combined with comprehensive support by image editing and viewing software and browsers also makes PNG an attractive alternative to TIFF and JPEG 2000 for an archival master format (Gillesse *et al.,* 2001).

According to Gillesse *et al.* (2001), the main disadvantage of PNG is that there is no option for *lossy* compression (other than decreasing the bit depth), resulting in relatively large file sizes.

2.4.5. GIF

Wiggins *et al.* (2001) state that GIF is the oldest and most widely supported Web-based graphic format. It was created in 1987 as an early solution to the problem of digital image storage. Its primary strength is its lossless LZW compression algorithm, which provides up to 4:1 lossless compression of images. Images converted to GIF are down-sampled from 24-bit (2^{24} = 16.7 million) colours to 8-bit (2^{8} = 256) colours (Donkin, 2001). The palette of 256 colours gives a good interpretation of the original image but the practical implication of this limitation is the inability of GIF to display an image with its full colour range (Wiggins *et al*., 2001).

While GIF files do not produce artifacts like JPEG files, tones that are graduated, like skies etc., will be posterised, where the continuous gradation of tone is converted to several regions of

fewer tones, which makes GIF undesirable for compression of photographic images and more suitable for illustrations and images with large areas of continuous colour.

The following section investigates various different large image support solutions.

2.5. Large image support solutions

There is very little literature available that deals directly with large image support within DRs, such as DSpace. Therefore, the focus will be on literature that deals with the concepts and technologies that form the basis of large image support solutions and also literature that documents solutions in commercial and non-commercial Internet applications.

Many Web applications (*Google Maps*[1], *Google Earth*[2], *Adobe Scene 7*[3], *Zoomify*[4], *Microsoft Deep Zoom*[5], etc.) provide large image support through the use of *zooming user interfaces* (ZUIs) (Bederson and Hollan, 1996). ZUIs provide an infinite canvas that can be accessed through panning and zooming controls (Liogkas & Tungare, 2002), as seen in Figure 3.

[1] Google Maps (http://maps.google.com/)

[2] Google Earth (http://earth.google.com/)

[3] Adobe Scene 7 (http://www.scene7.com/)

[4] Zoomify, Inc. (http://www.zoomify.com/)

[5] Microsoft Deep Zoom (http://livelabs.com/seadragon/)

Figure 3 - Zoomable User Interface

The images being accessed through a ZUI often are constructed from multiple derivative images or an image pyramid where the original large image is processed to produce 'tiles' of the image at different scales. Starting at 100% scale, the image is successively scaled in half to produce each tier until both the width and the height of the final tier are at most 256 pixels wide and 256 pixels in height, as seen in Figure 4 (Smith, 2007).

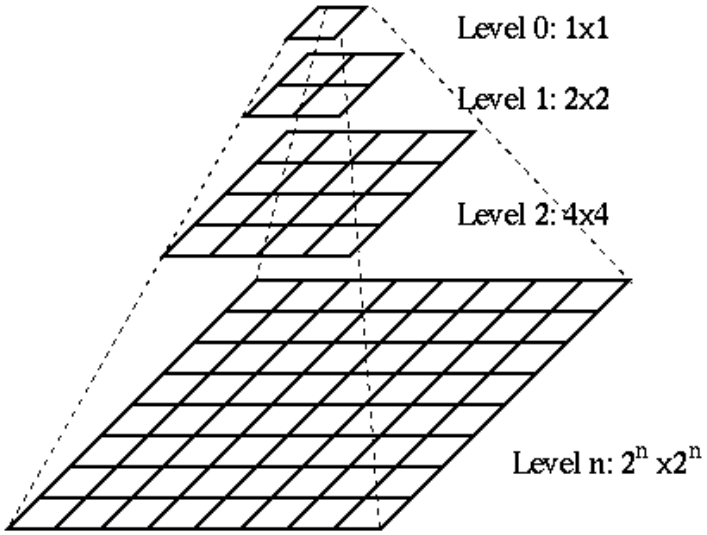

Figure 4 - Pyramidal Tile Organization

Tile organisation is pyramidal – stacked from a thumbnail down to the highest resolution, tier upon tier. Each tier's tiles are then saved in a separate directory. An XML file is also generated to serve as an index for the tiles. When the converted image is viewed, the Client Viewer uses the XML index to request tiles from the appropriate tier's directory to fill the display area. Only the tiles needed to stitch together the section of the image being viewed at a particular scale are requested. (Smith, 2007)

Each pan and zoom action causes a request for a small number of additional tiles – those that form part of the image panned to, at the level of zoom desired. These additional tiles are streamed on-demand, to the viewer. No tiles are therefore delivered unless required for the current display (*Zoomify Inc.,* 2003).

This method successfully limits the amount of data transferred to only match the region of interest requests and using a tile-based pyramid representation of the original image, the user is able to view the image in its original resolution even though he/she may be limited in screen real estate, by displaying only the region of interest of the original image. This tiling approach, however, requires the pre-generation and storage of each tile at all levels of resolution, which significantly increases storage requirements. At the client end, typical solutions use a JavaScript-driven HTML viewer (*Microsoft Seadragon*[6], *Google* Maps, *IIPImage*[7]) or a viewer based on a browser plug-in such as *Adobe's Flash Player*[8] (*Zoomify*, *Adobe Scene 7*).

[6] Microsoft Deep Zoom (http://livelabs.com/seadragon/)

[7] IIPImage (http://iipimage.sourceforge.net/)

[8] Adobe Flash Player (http://www.adobe.com/products/flashplayer/)

Adam Smith (2007) describes Zoomify Image, a Cornell University project in collaboration with *Zoomify* to create an open source, cross-platform and scriptable version of the processing software that creates the image data (image pyramid tiles and XML-index) for use in a *Zoomify* client viewer. He also describes how this project was implemented into a Web Content Management System to allow authors to upload high-resolution images which are then automatically processed on the server by Zoomify Image and then displayed within a *Zoomify* client viewer.

Ryan Chute and Herbert Van de Sompel (2008) introduce the Djatoka open source JPEG 2000 Image Server project as a solution for disseminating high-resolution images built around JPEG 2000 as the service format. So far, only commercial image servers with JPEG 2000 support have been available with steep license fees and limited extensibility capabilities. As an open source project, Djatoka provides extensibility under control of the community of implementers and developers. Djatoka takes advantage of JPEG 2000's multi-resolution image file format, which encapsulates a set of resolutions and dynamic region of interest extraction to successfully eliminate the need to store pre-generated tiles for all resolution levels.

In brief, Djatoka functions by resolving requests for a JPEG 2000 image by using the Kakadu library[9] to return a PNM (portable anymap) file to the Djatoka API. The Djatoka API then transforms the PNM file to the requested image format (default JPEG) and returns the image to the client.

Djatoka also was developed to support reuse through URI-addressability of all image disseminations, including regions, rotations and format transformations. It also supports dissemination for a range of image formats by dynamically creating the desired format from the JPEG 2000 master.

Djatoka not only provides disseminations for images that are stored locally on the image server, but URIs of any Web-accessible image can be passed to the Djatoka API as the image identifier. The Djatoka API will then obtain the image by dereferencing the URI and dynamically converting it to a JPEG 2000 file that is stored locally. At the client end, Djatoka provides an AJAX-based client, based on the *IIPImage* JavaScript Viewer, which allows panning, zooming and selecting the URI of the current view.

The creators of Djatoka expressed the need to further develop Djatoka to provide a rich open source client interface and to integrate it with popular repository platforms.

[9] Kakadu JPEG 2000 developer toolkit. (http://www.kakadusoftware.com/)

A commercial DSpace large image support module is available from *@mire,* a company that develops products and add-on modules for repositories (@mire, 2008). Their Image Zoom Module integrates with the DSpace platform to provide an image processing facility, which generates JPEG tiles at different resolution levels, which can then be accessed through a JavaScript viewer that offers panning and zooming functionality. The standard module supports JPEG and PNG images. Using the image Zoom Module in combination with @mire's Information Conversion Suite will enable JPEG2000, TIFF and GIFF support. After the Image Zoom Module installation, all new images inserted into the repository will become accessible through the client image viewer by default. The administrator can however limit the accessibility of each image by setting permissions on the image. The available permissions are: public (anonymous), private (registered users) and Intranet (within a certain IP range).

2.6. Concluding remarks

Various organizations around the world have started to implement DRs to manage, preserve and provide access to their ever-increasing collection of digital materials. Indeed, many Culture-based organizations have embraced this technology and are in the process of digitizing important collections of archives of printed materials, photographs and artwork. A factor that has helped spur this movement on is the emergence of freely available, open source DR packages that makes it possible for organizations to easily establish their own Digital Repositories.

In turn these DR platforms are still developing and evolving in response to the demands of the ever-increasing volumes of data, images, multimedia works and digital records and mass digitization efforts. The development focus has started to shift from building repositories to the delivery of services curtailed to the needs of end user.

Of the three DR platforms investigated, it was found that DSpace was the most popular amongst the various open source repository platforms. This may be due to the fact that DSpace provides all the functionality required to establish a long-term digital repository in a relatively simple fashion. Fedora, on the other hand, provides a more flexible architecture in the form of a set of Web services that enables it to act as a foundation layer for a variety of service oriented applications. This may make it less desirable for organizations looking for an out-of-the-box solution.

An evaluation of different image formats' suitability in terms of digital preservation activities revealed that three formats, TIFF, JPEG 2000 and PNG could be considered as candidates for long-term preservation formats because of their lossless compression algorithms and comprehensive support for metadata. The lack of browser software support for TIFF and JPEG 2000 makes these formats unsuitable for dissemination via the Web. JPEG format's compression algorithm effectively provides considerable cost-savings in terms of file size and its support for 24-bit full colour images makes it an excellent format for photographic image dissemination on

the Web. PNG and GIF on the other hand would be more suitable for disseminating illustrations and images that contain large areas of similar colour.

A popular approach to deliver large images over the Web is the use of ZUIs, which rely on an image pyramid to render a multi-resolution version of the image with pan and zoom functionality. The Cornell University project, Zoomify Image is an example of how this solution has been successfully implemented in a Web Content Management system. This project's weakness lies in the use of proprietary software (*Zoomify Client*) to enable viewing of large images. Many organisations would prefer to use an open-source solution that does not tie them to the development work of a single organisation.

The Djatoka open source JPEG 2000 Image Server project takes advantage of JPEG 2000's features such as multiple resolutions and region of interest extraction to successfully eliminate the need to pre-generate and store tiles for all resolution levels, which significantly reduces storage costs. This project however has not been implemented in a popular digital repository platform.

The availability of the commercial Image Zoom Module from *@mire*, highlights the need for a large image support system for digital repositories.

The following chapter outlines the research questions we aim to answer, and it describes the methods used to conduct our research to find a usable large image support solution for a popular digital repository platform.

3. Methodology

3.1. Significance of research

Online image collections are often built by storing uncompressed or lossless-compressed images. There are however, two real obstacles in terms of presenting these images to the end-user. First is the large file-sizes of these images, which could be hundreds of megabytes or even gigabytes in size, which cannot be transferred efficiently, even in high bandwidth networks. Second is the presentation of these large images, where the limited screen-resolution of the end user's monitor only allows a small portion of the original image at its original resolution to be displayed.

As introduced in the background chapter, there are two methods that effectively overcome these obstacles to deliver a large image to the user in its original resolution. The first solution relies on a zooming user interface (ZUI), which presents an infinite canvas and provides panning and zooming controls to inspect a large image. Images being accessed through ZUI applications are often constructed from multiple derivative images or an image pyramid where the original large image is processed to produce 'tiles' of the image at different resolutions. This method requires the pre-generation and storage of the image pyramid tiles. The second method relies on accessing a single compressed image, which can deliver multiple resolutions of the original image within a single code-stream. JPEG 2000 supports this feature but requires proprietary software to encode or compress an image. Another disadvantage of JPEG 2000 is the lack of browser support for this standard. None of the popular browsers has built-in support for the JPEG 2000 format. For this reason, a decision was made to continue this research using the multiple derivative image pyramid approach that is accessed with a ZUI.

Digital Repositories (DRs) are often used to construct online image collections but DRs do not have built in functionality to deliver large images effectively over the Internet without relying on custom-built or commercial solutions.

This research therefore investigates the possibility of integrating a large image solution in a popular digital repository. A solution that is based on leveraging ZUI and image pyramid technologies coupled with the necessary tools to generate the image pyramid' multiple derivative images.

3.2. Research questions and hypotheses

The following questions (in **bold font**) drove the research design and the anticipated answers (in *italic font*) were used to formalize the research hypotheses.

1. **Are there solutions available that deal effectively with the problems of delivering large images over the Internet?** There are many examples of Web applications (*Google*

Maps, *Google Earth*, *Adobe Scene 7*, *Zoomify*, *Microsoft Deep Zoom*) that successfully leverage ZUIs that access multiple derivative images from a image pyramid to display images at various resolutions and provide panning and zooming controls to navigate these images.

H1: Effective solutions exist to provide large image support to Web applications.

2. **Do digital repositories provide support for delivering large images over the Internet?** Digital repositories provide comprehensive support for various file formats but no additional support for delivering large images over the Internet.

   H2: Digital Repositories do not provide large image support.

3. **Is it possible to incorporate an effective large solution within a popular Digital Repository?** Investigation of existing Web-based large image solutions could help provide a method to provide a popular digital repository with additional support for delivering large images over the Internet.

   H3: It should be possible to incorporate existing large image support within a popular Digital Repository.

3.3. Methods

3.3.1. Large Image Support Survey

Many commercial and non-commercial Web applications can successfully deliver large images over the Internet. *Google Earth* and *Google Maps* are popular examples of how the ZUI and multiple derivative images approach allows a user to seamlessly inspect images that are effectively gigabytes in file size at various resolutions.

Many organizations have used DRs to build online image collections (Tedd and Large, 2005). Some of these organizations have chosen to purchase commercial large image support solutions (@mire, 2008) while others have developed custom large image support solutions that have been integrated in their digital repository systems (Northwestern University Library, n.d.).

How do these custom solutions compare with the solutions that exist beyond the scope of DRs? To answer this question, a survey was constructed to gather user feedback data with regards to the usability and efficacy of current large image support solutions that have been implemented within digital repositories (DRs) and also commercial solutions that exist beyond the scope of digital repositories.

The survey focused on four different large image solutions. Two of these solutions represented commercial applications while the other two represented custom large image solutions that have been implemented within DR systems.

The results of this survey were used to inform the design and development of the prototype systems.

3.3.2. Procedure

This research is focused on human-computer interaction (HCI). Sharp *et al.* (2007) define interaction design as "designing interactive products to support the way people communicate and interact in their everyday working lives". According to Norman's evaluation/execution model, a good interactive system is one where a user can easily work out how to operate the system in an attempt to achieve his goals and also easily evaluate the results of his action on the system (Norman, 1988). Jones and Marsden (2006) note that HCI projects involves three main types of activity (Jones & Marsden, 2006):

- **Understanding users** – having a sense of people's capabilities and limitations; gaining an insight into their lives, the things they do and use.
- **Developing prototype designs** – representing a proposed design in such a way that it can be demonstrated, altered and discussed.
- **Evaluation** – using evaluation techniques to identify the strengths and weaknesses of a design.

These three activities formed the basis for development of this project and will be discussed in the following sections.

3.3.3. Understanding users

The goal of this project is to provide a solution that will support the users of a digital repository to submit and view large images, specifically users of a DSpace repository. Therefore it is necessary to develop a profile of the average DSpace user that will perform the required tasks that enable large image support within DSpace.

Dana McKay's (2007) summary of the literature on the usability of digital repositories suggests that DR users can be divided into three main groups: authors, information seekers, and data creators/maintainers (henceforth data maintainers). Authors and data maintainers are primarily responsible for submitting material into the DR. To assist with the submission of material, DSpace offers a flexible, easy-to-use submission process. Submitters simply need to complete a brief submission form and grant permission to distribute and preserve the work. The technical requirements in terms of submitting material are therefore quite minimal.

In terms of information seekers, McKay notes that there are very few available usability studies of DRs but, despite this lack, an understanding of how users are likely to use DR for information seeking can be gained by investigating studies of journal databases and open access research repositories. She points out that while these studies do not investigate DR usage, the systems they describe are similar enough in purpose to DRs and that information seekers will use them in similar ways. Given that assumption she found that typical DR users will visit infrequently, download only a few articles at a time, perform very simple searches, and use results from the top of the results list (though they will browse widely in other ways if offered the chance).

Therefore, in terms of submission and accessing of materials, a typical user-profile for a DR such as DSpace does not require any specific technical abilities other than an average computer literacy. The selection process of participants for the survey and evaluation studies therefore was focused on selecting people that possessed an average or above average level of computer literacy. Requests for participation in the surveys and evaluation studies were sent to the *DSpace General Mailing List*, the *UCT Computer Science Digital Labaratory Mailing List* and to a personal network of friends and colleagues working in the field of online marketing and Web development.

3.3.4. Design and prototypes

Providing DSpace with a large image support solution could be accomplished either by modifying the DSpace platform or by developing an independent system to provide large image support without any modification of the DSpace source code. In the previous section this project's target users was identified. To make the system usable to this target audience it was imperative to keep the technical barriers of usage very low. In their usability study of DSpace installation and configuration, Körber and Suleman (2008) found that most users experienced significant problems during the DSpace installation process. The technical barriers of DSpace installation and even DSpace recompilation was deemed too high and a decision was made to avoid any solutions that required modification to the DSpace source or development of an add-on module that would require a recompilation to integrate it within the platform.

As introduced in the background chapter, there are two methods that effectively overcome the obstacles of restricted screen real estate and bandwidth limitations to deliver a large image to the user in its original resolution. The first solution relies on a zooming user interface (ZUI), which presents an infinite canvas and provides panning and zooming controls to inspect a large image. Images being accessed through ZUI applications are often constructed from multiple derivative images or an image pyramid where the original large image is processed to produce 'tiles' of the image at different resolutions. This method requires the pre-generation and storage of the image pyramid tiles. The second method relies on accessing a single compressed image, which can deliver multiple resolutions of the original image within a single code-stream. JPEG 2000 supports this feature but requires proprietary software to encode or compress an image. Another

disadvantage of JPEG 2000 is the lack of browser support for this standard. None of the popular browsers has built-in support for the JPEG 2000 format. For this reason, a decision was made to continue this research using the multiple derivative image pyramid approach that is accessed with a ZUI.

These design decisions were implemented as a prototype to provide a simplified, concrete implementation of the digital repository large image support solution. The development of the prototype system would adhere to the principles of user-centered design (UCD). In UCD, evaluation by users form part and parcel of the prototype development, where the users' suggestions for improvements are implemented in the iterative process of prototyping (Van House *et al*., 1996).

Jones & Marsden (2006) discussed the two stages of prototyping: *low-fidelity* and *high-fidelity*. A low-fidelity prototype is one that uses materials that are very different from the finished product; its purpose is to be cheap, simple and quick to manufacture (Sharp *et al.*, 2007). The purpose of low-fidelity prototyping is to help conceptualize a design and to investigate new ideas. It is often used early in the design process before exact details are known. High-fidelity prototyping employs materials that are expected to be in the final product and produce a prototype that resembles the final product (Sharp *et al.*, 2007).

For the purpose of this research the development of a low-fidelity prototype was deemed to be unnecessary, as most of the details of the design have already been identified. The goal is to create high-fidelity prototypes by implementing existing ZUI and image pyramid technologies as building blocks. A large portion of prototype development is focused on making intelligent compromises about what to test and what not to test (Jones & Marsden, 2006). These compromises lead to two types of prototypes: *horizontal* and *vertical*. A horizontal prototype typically provides a wide range of functionalities but with little detail, whereas a vertical prototype only has a few functions but provides a lot of detail (Sharp *et al.*, 2007). Providing DSpace with large image support requires only a few functions: image processing, DSpace import and image access. Therefore vertical prototyping was adopted to develop the high-fidelity prototypes. The prototype implementations were then evaluated.

3.3.5. Evaluation

The goal of evaluation is to assess how well a design (or prototype) fulfilled its intended purposes. Evaluations are used to test whether the product can be used effectively, efficiently and with satisfaction. Users require products that provide an agreeable and engaging experience (Sharp *et al.*, 2007). Evaluation may therefore be a process to verify user experience with a product.

The goal of this research project is to incorporate effective large image support within the DSpace platform. The measure for evaluating the success of this project is the usability of the system in terms of providing DSpace with large image support.

According to Dumas and Redish (1999), usability means that people can use a product, quickly and easily, to accomplish their own tasks. Their definition rests on four points:

1. Usability focuses on users.
2. People use products to be productive.
3. Users are busy people trying to accomplish tasks.
4. Users decide when a product is easy to use.

Therefore, to measure usability, it is important to understand and work with people who represent the actual or potential users of the product. People consider a product "easy to learn and use" in terms of the time it takes to do what they want. People connect usability with productivity *i.e.* how well the system supports them in being productive.

Jones and Marsden (2006) identified a list of evaluation methods that may be used to test prototypes, from low-fidelity to high-fidelity prototypes. Some of these methods require participation from end-users and others do not. For our purpose, we required a method that included end-users to test whether or not the system was usable and to determine problems and potential solutions/improvements.

User testing typically requires users to complete a set of specific tasks while performance, error rates and user satisfaction are measured. Usability testing requires relatively few participants - quick and dirty tests may involve only one or two subjects, otherwise 6 to 12 users is the recommended sample size (Dumas and Redish, 1999).

The method adopted for evaluating the prototype implementations is a combination of *direct observation* within a controlled environment and *interviews* to obtain qualitative results and feedback. Interviews provide the opportunity to ask a subject some questions about their actions during the experiment, which can lead to interesting insights into how their mental model diverges from yours (Jones & Marsden, 2006).

Evaluation method and procedure

To assess the effectiveness of the prototype system, participants were recruited to perform the steps that authors and data maintainers would have to follow to enable large image support for DSpace.

The participants had to follow a list of instructions that would take them through the steps of processing a folder of large images, importing the generated assets into DSpace using the DSpace batch importing command-line utility and, finally, viewing a large image via the client viewer on a Web page hosted on the DSpace server. The instruction list did not include the administration steps required to create a collection in DSpace. Evaluation of these steps falls beyond the scope of the prototype evaluation study.

The participants were observed while they performed each tasks on the instruction list. Notes were made when a participant failed to complete particular task. The time that each participant took to complete the entire process was also measured. After the evaluation the participants were given the opportunity to provide comments or feedback.

The problems observed and the feedback gathered from the first user-based evaluation was then analysed to make design decisions to improve the prototype implementation.

Finally, tests were performed to evaluate the system's image processing and batch importing speed. The amount of data transferred while inspecting a high-resolution image was also measured to test whether the prototype system was able to deliver large images in a bandwidth-conscious manner.

Evaluation participants

For the first prototype evaluation, twenty-four subjects were recruited to participate in this study. All recruited subjects were adults. Fifteen were male and nine were female. Their ages range between 19 and 39 years old. All of the participants were professionals working in the fields of Web-development, online marketing and copy writing. None of the participants had prior knowledge of DSpace or large image solutions.

To evaluate the second prototype system another group of twenty-four people were invited to partake in the user-based study. As with the first group of participants, the participants were professionals working in the fields of Web-development, online marketing and copy writing. 15 were female and 9 were male with their ages ranging between 19 and 45 years and none of the participants had experience with DSpace or large image solutions.

Performance testing

To measure the performance of the prototype, tests were performed with the final prototype to evaluate the system's image processing and batch importing speed. The amount of data transferred while inspecting a high-resolution image was also measured to test whether the prototype system was able to deliver large images in a bandwidth-conscious manner.

3.4. Constraints and limitations

This section identifies the constraints and limitations of this project.

Materials for prototypes – large (high-resolution) images are required to test the prototype implementations. These images must not violate copyright. Furthermore, the images must be large enough to sufficiently test the prototype implementation. For the purpose of this project, we define large images as images with a resolution of higher than 5000 x 5000 pixels.

Equipment and software (prototypes) – Developing new methods of large image support falls beyond the scope of this project. Instead the goal is to leverage existing solutions (software) to provide large image support to DSpace. To build and test the prototype we require a computer with a working installation of a current DSpace release.

Availability of users – The profile of DSpace users was discussed in an earlier section. Due to the lack of availability of existing DSpace users, people were recruited for the survey and user-based evaluations from the *DSpace General Mailing List*, the *UCT Computer Science Digital Laboratory Mailing List* and colleagues working in the field of online marketing and Web development.

3.5. Concluding remarks

The goal of this project is to investigate the possibility of integrating a large image solution in a popular digital repository. A solution that is based on leveraging existing ZUI and image pyramid technologies coupled with the necessary tools to generate the image pyramid' multiple derivative images. To support this research the following methodology was adopted. First, a survey was conducted to gather comparative data about custom DSpace large image support solutions and commercial large image support solutions. The survey data was then used to inform the decision making process while developing the prototype large image support solution. The prototype development follows an iterative process of prototyping that adheres to the principles of UCD where evaluation by users form part and parcel of the prototype development and users' suggestions for improvements are implemented in the iterative process.

The following section provides a report of the large image support survey.

4. Large Image Support Survey

An online Large Image Support Survey was conducted with the goal to rate the effectiveness of four different Large Image Support solutions. Two of these solutions represented commercial applications that did not directly form part of DRs while the other two represented solutions that have been implemented within DR systems. The commercial solutions were *Zoomify* (Zoomify Inc, 2003) and Microsoft's *Seadragon* (Microsoft, 2010). The two solutions integrated within digital repositories were the *West Texas Digital Archives (WTDA),* which implemented *@mire's* commercial Image Zoom Module (Abilene Library Consortium, n.d.) and the *Northwestern University Library's (NWUL)* experimental large image viewing solution (Northwestern University Library, n.d.). It should be noted that these systems were selected for the survey because they shared similar technologies in terms of their client interface. Microsoft *Seadragon* and *WTDA* both employed an AJAX-based, JavaScript client implementation whereas the *Zoomify* and *NWUL* solutions both employed a Flash-based client interface.

4.1. Survey Questions

The survey questions were developed with the purpose of getting ratings on how effective the various solutions were in presenting large images to the end-user.

There were two types of questions: Simple Yes/No and a 5-point rating question. The rating questions consisted of the following options: Excellent (5), Good (4), Average (3), Below Average (2) and Poor (1). The same questions were asked in the same order for all 4 Large Image Support solutions. The questions are listed below:

1. Were you able to zoom-in on the image from "XXX" Website? (Yes/No)

2. Did you need to install additional software to be able to zoom in on the image?

3. Rate how successful you were in zooming-in on the image:

4. Rate the control mechanisms that enable you to zoom-in on the image:

5. Rate the feedback mechanisms provided by the zooming function:

6. Rate the response speed of the zooming functionality:

7. Rate the effectiveness of "XXX's" Large Image Support:

4.2. Survey Participants

Requests to partake in the survey were sent to the *DSpace General Mailing List*, the *UCT Computer Science Digital Labaratory Mailing List* and to a personal network of friends and colleagues working in the field of online marketing and Web development.

A total of 29 people responded to these requests, which resulted in 24 completed surveys. The results of the survey are summarised below. The complete results, comments and feedback can be found in Appendix A.

4.3. Survey Results

The results of each question were plotted on a chart to graphically illustrate the results. The total number of people that chose a certain option or rating is plotted along the vertical axis. The 4 zooming solutions were plotted on the horizontal axis for the first 2 questions where the yes/no answers are represented by a unique color. For questions 3-7, the ratings (Excellent, Good, Average, Below Average and Poor) were plotted along the horizontal axis where each zooming solution is represented by a unique color.

Figure 5 charts the results for Question 1: Were you able to zoom-in on the image. Of all the zooming solutions, only *Zoomify* did not receive 100% yes answers. One person indicated that he/she could not zoom-in on the image. These results therefore, indicate that all the solutions provided usable zooming functionality.

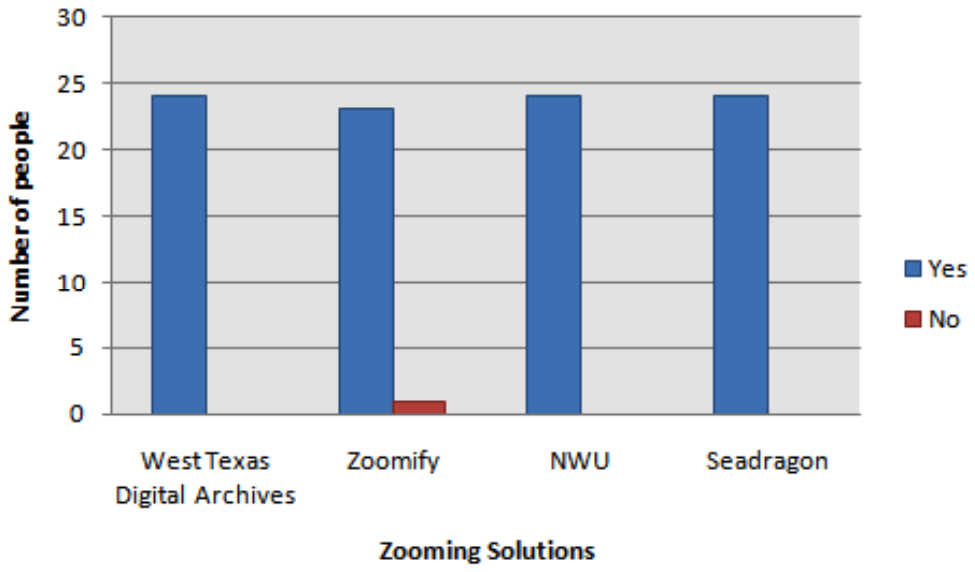

Figure 5 - Question 1: Were you able to zoom-in on the image

Figure 6 (Question 2) shows that the vast majority of respondents did not need to install additional software to be able to zoom-in on the image. *Zoomify* required 2 respondents to install additional software. This is probably due to the fact that the Zoomify client is Flash-based and therefore requires Adobe's Flash Player browser plug-in. *Seadragon* required one person to install additional software, which is interesting because it is a JavaScript-based client.

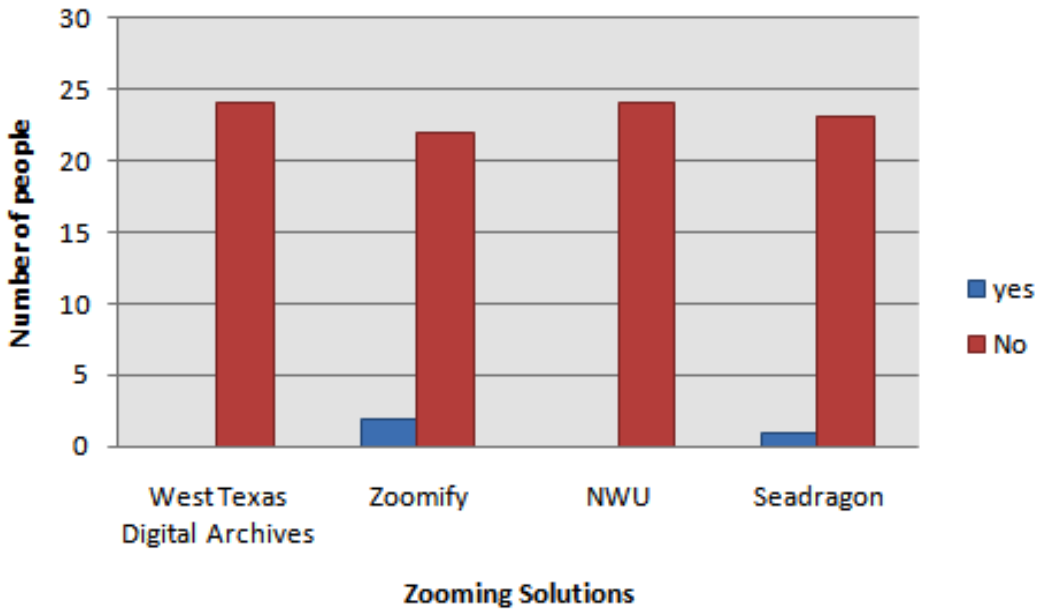

Figure 6 - Question 2: Did you need to install additional software to be able to zoom-in on the image

Figure 7 (Question 3) charts the ratings of how successful the respondents were in zooming-in on the image according to the different zooming solutions. This chart clearly illustrates that the two commercial solutions, *Zoomify* and *Seadragon,* received much better ratings than the non-commercial solutions. It is noteworthy that 92% or 22 out of 24 respondents rated the *Seadragon* and *Zoomify* solutions in the Good to Excellent range.

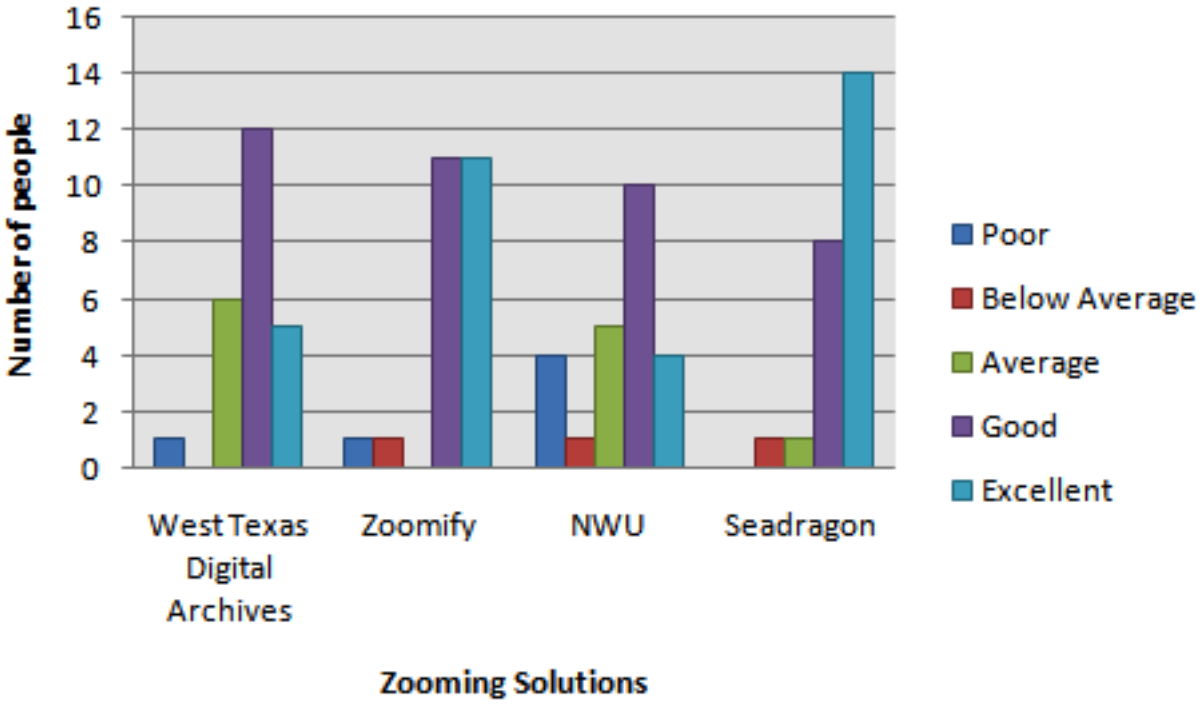

Figure 7 - Question 3: Rate how successful you were in zooming-in on the image

Figure 8 (Question 4) illustrates that the respondents found the control mechanisms that enable you to zoom-in on the image from *Zoomify* and *Seadragon* superior to those from *NWUL* and *WTDA*. *Seadragon* received the best ratings with slightly more than half (54 %) or 13 people giving it an excellent (5/5) rating.

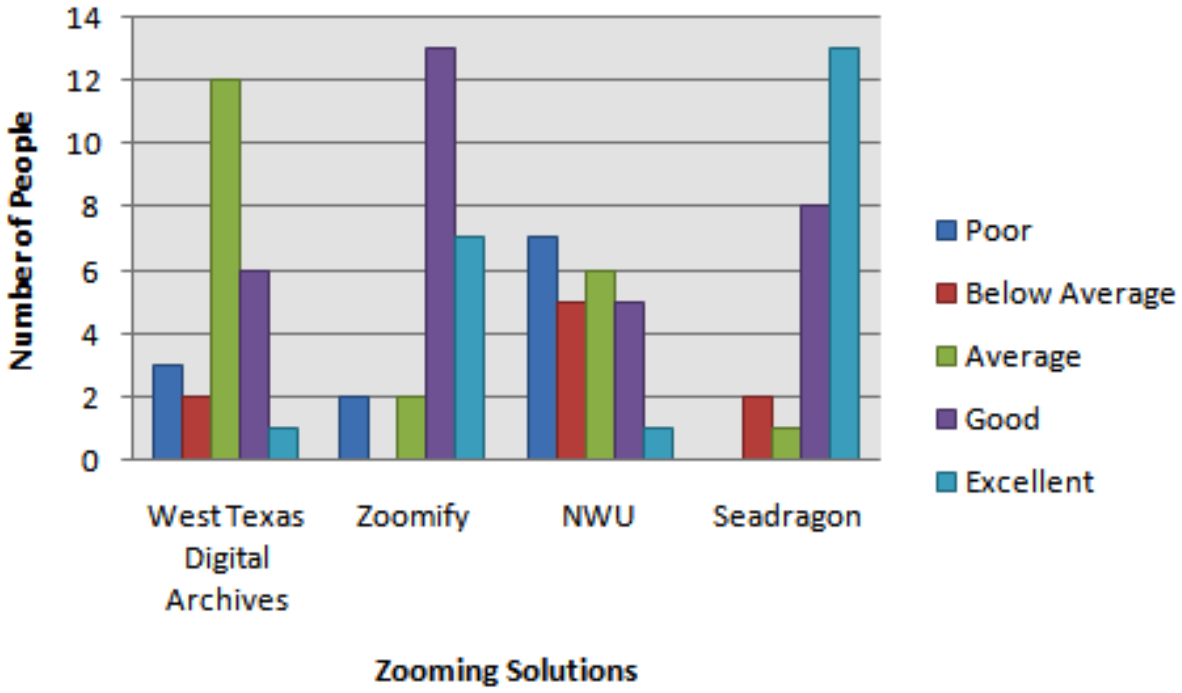

Figure 8 - Question 4: Rate the control mechanisms that enable you to zoom-in on the image

Figure 9 (Question 5) shows that the feedback mechanisms provided by *Zoomify's* zooming function were superior to the other solutions' feedback mechanisms with a total of 87.5% or 21 respondents rating it in the Good to Excellent range. *Seadragon* also received a majority of positive responses with 75% or 18 people rating it in Good the Excellent range.

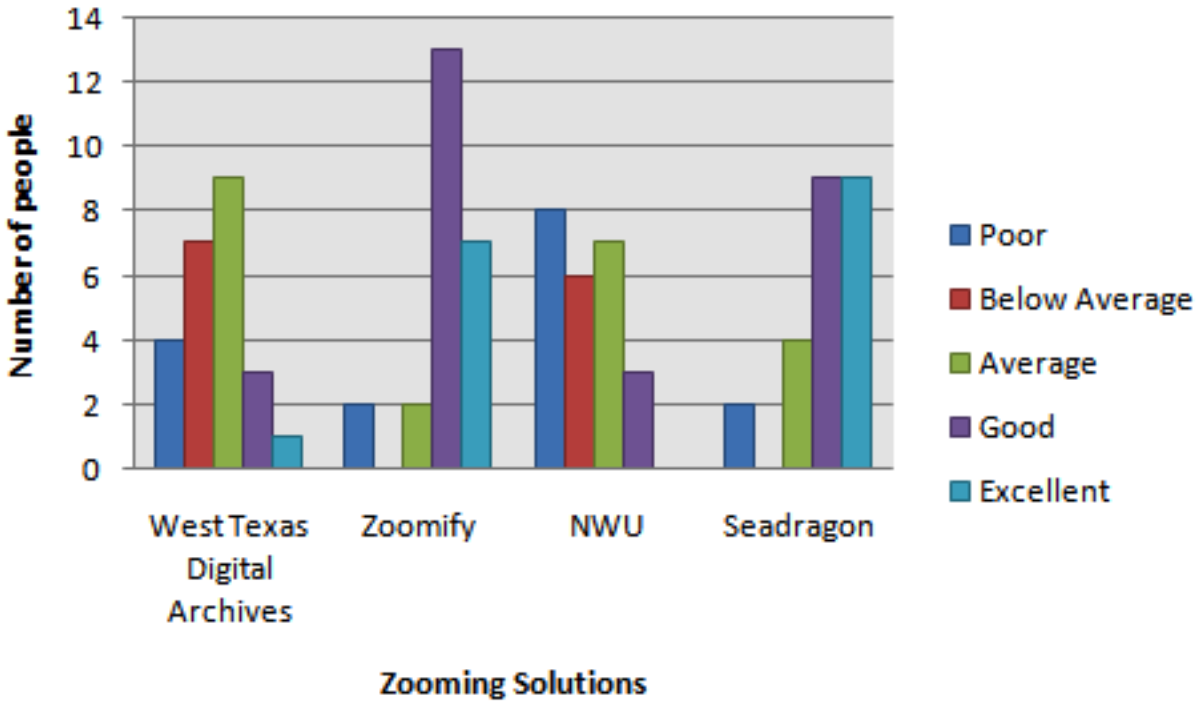

Figure 9 - Question 5: Rate the feedback mechanisms provided by the zooming function

The chart in Figure 10 (Question 6) shows that the people found the response speed of, *Seadragon's* zooming functionality to be much better than the other solutions with 71% or 17 of the respondents giving it an Excellent (5/5) rating.

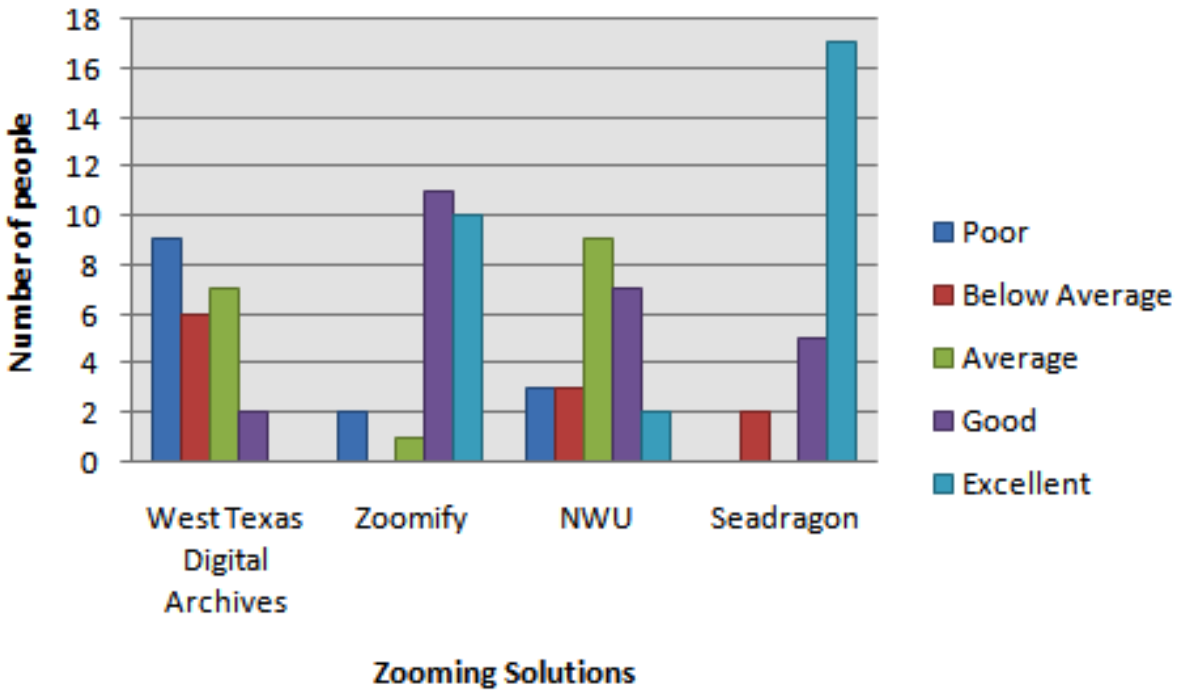

Figure 10 - Question 6: Rate the response speed of the zooming functionality

Figure 11 charts the final question's results where the respondents were asked to rate the overall effectiveness of these large image support solutions. The two commercial solutions - *Zoomify* and *Seadragon* - again dominate the good and excellent ratings with *Seadragon* receiving the best results with 92% or 22 people rating it in the Good to Excellent range.

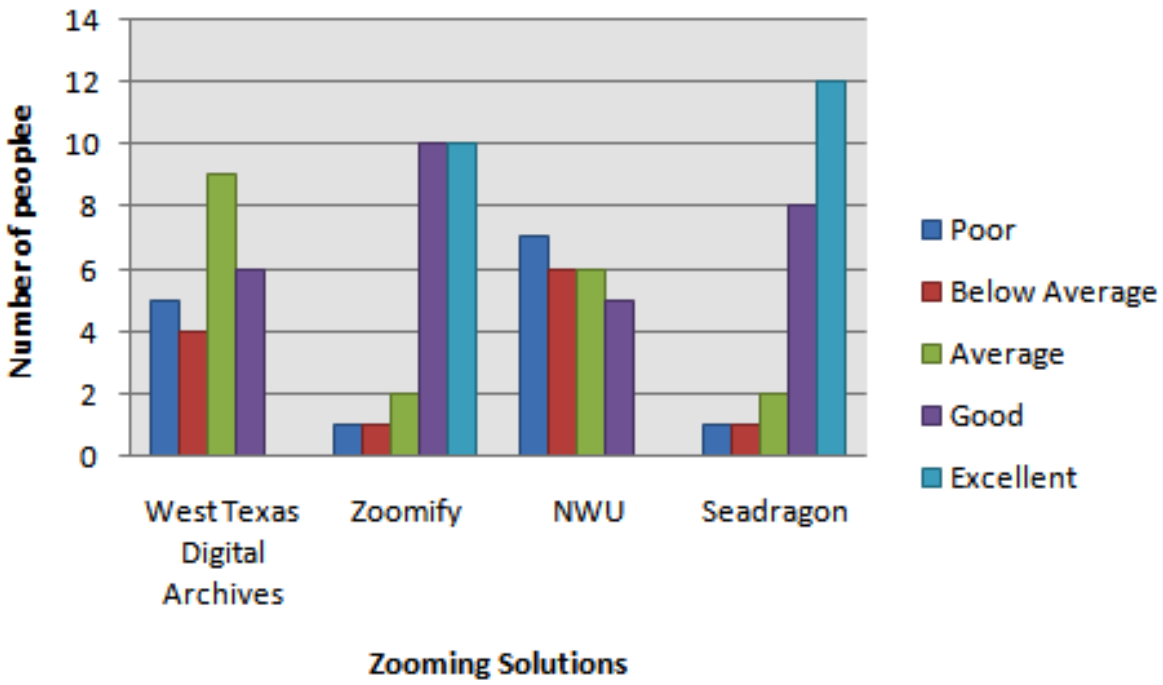

Figure 11 - Rate the effectiveness of Large Image Support

4.4. Concluding remarks

The figures in the previous section illustrate that the respondents found the large image support solutions from *Zoomify* and *Microsoft Seadragon* to be superior to the two solutions that were implemented within Digital Repositories. The solutions from *West Texas Digital Archives* and *Northwestern University Library* show a more even distribution of results with the majority of the ratings ranging between Good and Poor. Results for these solutions never dominated the Good and Excellent ratings.

The commercial solutions, however, leave room for improvement if we take the comments into account. *Zoomify's* sliding control received positive feedback although the interface lacked a maximize functionality. *Seadragon* did have a maximize functionality but lacked the sliding zoom controller. Two people also commented that a navigator or mini-map feature, similar to the one situated on *Zoomify's* interface that gives feedback of where you are in terms of the original image would make *Seadragon's* solution the best.

The following chapter presents the design and implementation of the prototype large image support solution.

5. Prototypes

This chapter discusses the development of two prototype large image support solutions.

5.1. Prototype goals

The main goal in developing the large image support prototypes was to produce software that will enable large image support within a popular digital repository system, such as DSpace. This includes:

1. The development/integration of a client-side large image-viewing module.
2. The facility to batch-process large images in order to generate the required assets (image tiles, client-side viewer, XML files, etc.) that enable large image viewing over the Internet.
3. The facility to incorporate these features into *DSpace*.

The aim was to develop a prototype with as little modification to the *DSpace* platform as possible and therefore employ an approach where the solution had to be moulded around *DSpace's* existing architecture.

5.2. First high-fidelity prototype

5.2.1 Tools employed

The first prototype implementation was built by integrating and modifying the following software tools/packages:

- *OpenZoom*[10] platform - *OpenZoom* is a *Flash*[11] or *Flex*[12] based open source large image support platform. *OpenZoom* provides all the tools required to build a client-side flash-based viewer that can be modified extensively.
- *DeepZoomTools.dll*[13] – Developed by *Microsoft* for use in their *DeepZoom* project. DeepZoomTools.dll provides the required methods to generate the image tiles.
- *Dspace Batch Import Module*[14] – Included with the *DSpace* package, it provides a command-line interface for batch item importing.

[10] Open Zoom (http://openzoom.org/)

[11] Adobe Flash CS4 Professional (http://www.adobe.com/products/flash/)

[12] Adobe Flex 3 (http://www.adobe.com/products/flex/)

[13] DeepZoomTools.dll (http://blogs.msdn.com/expression/archive/2008/11/26/hello-deepzoomtools-dll-deep-zoom-image-tile-generation-made-easy.aspx)

[14] DSpace Release 1.6.0 Item Importer Code (http://dspace.svn.sourceforge.net/viewvc/dspace/trunk/dspace-api/src/main/java/org/dspace/app/itemimport/ItemImport.java)

5.2.2. Client-side viewer

The Flash-based client viewer was developed by modifying the source code of the *OpenZoom* package.

The *OpenZoom* package provides the source code for a variety of platforms, including *Flash CS3*, *CS4* and *FlexBuilder,* to compile a Flash-based large image viewer.

When compiled it produces two files: the Flash-based viewer, MultiScaleImage.swf; and an HTML file that includes the embed object code to display MultiScaleImage.swf on the HTML page.

The prototype viewer was developed by modifying the standard *OpenZoom* viewer in three ways. First, the user-interface was modified to include buttons that enable zooming in and out of the image and a 'show all' button to reset the image to the starting resolution (see Figure 12).

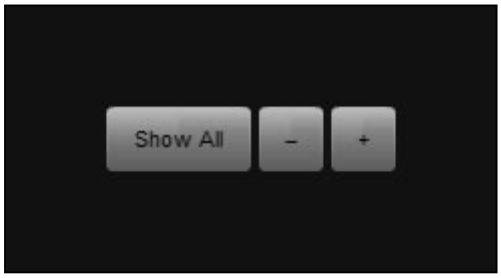

Figure 12 - Buttons to enable zooming in and out of the image

Second, the source code was modified to read dynamic image data. The standard *OpenZoom* viewer reads image data that is hard-coded in the source code. Therefore, the code was modified to read image data via the *flashvars* parameters in the HTML document using the *parameters* property of the *LoaderInfo class*[15].

Third, the source code was modified to read image tiles from a flat directory structure. *OpenZoom's* default behavior is to fetch the image tiles from a directory structure where each tier's tiles are saved in a separate folder. *DSpace*, however, requires that an item and its related files be imported in a flat directory structure. For example, to import a website as an item in *DSpace*, all the website's files – the HTML, images and JavaScript files needs to be in one folder.

5.2.3. Image processing

In *DSpace*, data is organised to reflect the structure of the organisation using the *DSpace* system[16]. Each *DSpace* site is divided into *communities*, which can be further divided into *sub-communities* reflecting the typical university structure of college, department, research center or laboratory.

Communities contain *collections*, which are groupings of related content. A collection may belong to more than one community.

Each collection is composed of *items*, which are the basic archival elements of the archive. Each item is owned by one collection. An item may appear in other collections, but each item only has one owning collection.

Items are further subdivided into named *bundles* of *bitstreams*. Bitstreams are, as the name suggests, streams of bits, usually ordinary computer files. Bitstreams that are somehow closely related, for example HTML files and images that compose a single HTML document, are organised into bundles.

---

[15] Adobe documentation: LoaderInfo Class
(http://livedocs.adobe.com/flex/3/langref/flash/display/LoaderInfo.html - parameters)

[16] DSpace System Documentation: Functional Overview
(http://scm.dspace.org/svn/repo/dspace/tags/dspace-1.5.2/dspace/docs/html.legacy/functional.html)

To build the prototype application, an executable Windows Form program was developed in C# to batch process a folder of large images in order to generate the required *bundle* of *bitstreams* (image tiles, client-side viewer, XML files, etc.) that would provide large image support for the *DSpace* platform.

The core functionality of processing large images to generate the image-pyramid tiles is provided by Microsoft's *DeepZoomTools.dll*, which is installed as part of Microsoft's *Deep Zoom Composer*[17] application.

The output of the *DeepZoomTools.dll* methods follows the standard format where a directory is created that contains the image data XML file and the image-tile folders where all the tiles are grouped in a separate folder for each tier of the image pyramid. The naming convention of the generated image tiles is illustrated in Figure 13.

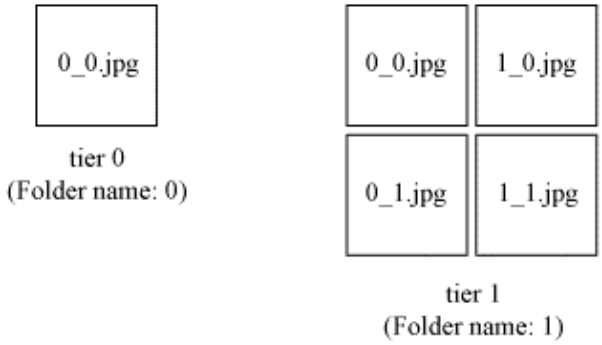

Figure 13 - DeepZoomTools.dll image tile naming convention

[17] Deep Zoom Composer (http://blogs.msdn.com/expression/archive/2009/02/18/deep-zoom-composer-february-2009-preview-released.aspx)

Bundles in *DSpace* must consist of a flat directory structure. Therefore, the prototype application modifies the output of the *DeepZoomTools.dll* operations into a flat directory structure, where all the tiles are located in one folder. The image-tiles are renamed as illustrated in Figure 14.

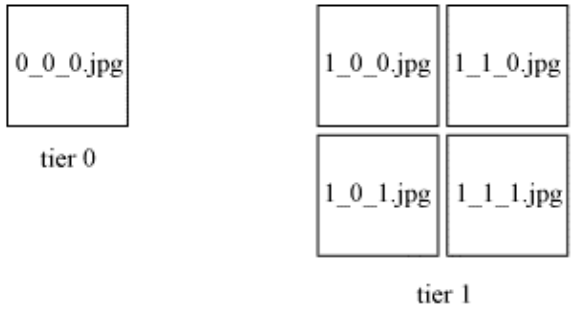

Figure 14 - DSpace image tiles naming convention

The application also generates an HTML file that contains a reference to the image data XML file via the *flashvars* parameters, which the client viewer (MultiScaleImage.swf) uses to locate the image tiles. The HTML also contains the embed object code to embed the client viewer on the HTML page. The client viewer file is then copied to each image item folder.

For security reasons, a Flash movie playing in a Web browser is not allowed access to data that resides outside the exact Web domain from which the SWF originated[18]. Any attempt to load content into a SWF file at runtime is subject to the Flash Player security model, which is in place to protect users and website owners.

[18] Adobe Technote External data not accessible outside a Flash movie's domain (http://kb2.adobe.com/cps/142/tn_14213.html)

This security model was set up to parallel the default settings provided in most Web browsers. As part of this model, the Flash Player, by default prevents cross-domain loading of data, but allows cross-domain sending of data. Since the release of Flash Player 7, a cross-domain policy file can grant the Flash Player access to data from a given domain without displaying a security dialog. When placed on a server, it tells the Flash Player to allow direct access to data on that server. The server can be in any location available to the Flash movie and does not have to be in the same domain. Cross-domain policy files, named crossdomain.xml, are placed at the root level of a server. Therefore, to prevent the prototype application from displaying security dialogues while accessing image data, a cross-domain policy file is also copied to each image item's folder to enable the client viewer to access image data from the hosting domain server.

DSpace requires that each item must have one qualified Dublin Core metadata record[19] for every item, for interoperability reasons and ease of discovery. Therefore, for each image item, a Dublin Core metadata file is generated and saved within each image-item folder.

The *DSpace* batch importing module has three specific requirements that had to be met by the prototype application to enable successful batch importing of the image.

1. A qualified Dublin Core metadata file to be included for each item.

2. A separate mapfiles folder[20] to accompany all the items to be imported. *Dspace* use this folder to generate the details of all items that were imported and their new location within the system—this file can help with future exports or removal of groups of imported content. The presence of this folder is required before launching the batch process.

3. Finally, it also requires that a contents file be included in each item's folder. It is a simple text document that lists each file that an item consists of, on separate lines. Therefore, in this case a text file is generated and saved in each image-item's folder that list all image-tile file names, the HTML file, the MultiScaleImage.swf file, the image-data XML file, the Dublin Core metadata XML file as well as the crossdomain.xml file.

[19] DSpace System Documentation: Functional Overview (http://scm.dspace.org/svn/repo/dspace/tags/dspace-1.5.2/dspace/docs/html.legacy/functional.html)
[20] DSpace System Documentation: Functional Overview (http://scm.dspace.org/svn/repo/dspace/tags/dspace-1.5.2/dspace/docs/html.legacy/functional.html)

Figure 15 shows the contents of a typical item folder generated by the prototype application.

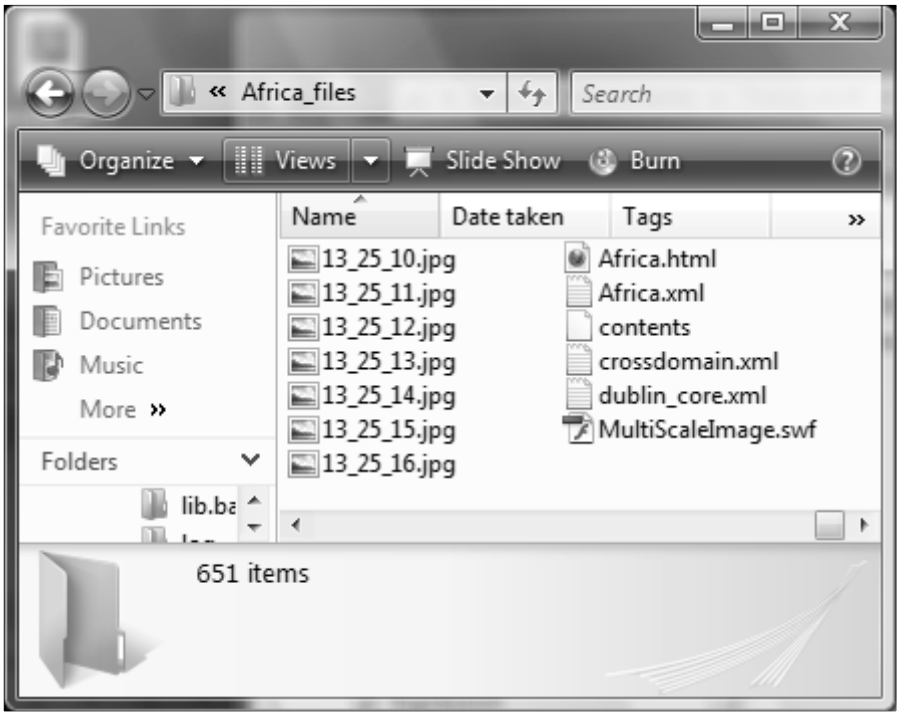

Figure 15 - Contents of a typical item folder created by the prototype application

5.2.4. DSpace batch importing

DSpace provides a command-line utility for batch importing of items. As mentioned in the previous section, this utility has strict requirements that need to be met to enable successful batch importing.

After all the items have been imported, the HTML file for each item needs to be set as the primary bitstream[21] which would cause *DSpace* to serve the HTML file when a request is made to view the large image.

[21] DSpace Tech *Import tool: How to set primary bitstream* (http://www.mail-archive.com/dspace-tech@lists.sourceforge.net/msg07655.html)

5.2.5 Summary of first prototype

The first prototype system was developed by modifying the standard *OpenZoom* client viewer to include buttons to enable zooming, panning and the ability to reset the image to its starting resolution. Further modification of the *OpenZoom* platform was required to enable reading dynamic image data from an HTML file.

An executable Windows Form program was built to batch-process a folder of large images. *Microsoft's DeepZoomTools.dll* methods were employed to produce the image-pyramid tiles and image data or index files. The outputs of these methods were reformatted to prepare them for batch ingest into *DSpace*.

Finally, the primary bitstream is set in the contents file of each item to ensure that DSpace serves the html file containing the client viewer when a request is made to view the image.

5.3. First prototype evaluation

A User-based evaluation study of the prototype was conducted to assess the effectiveness of the prototype DSpace large image support system. The goal of the study was to ascertain whether the prototype system was successful and usable in providing large image support to the DSpace platform.

The complete results, comments and feedback of the user-based evaluations of the prototypes can be found in Appendix B.

5.3.1. Evaluation instruction list

1. Image Processing

1.1 In the open window double click on DeepZoomer.exe

1.2 In the "Browse For Folder" dialogue window - select Computer, then Local Disk (C:), then Maps, then select the OK button.

(The Browse for folder dialogue window will disappear - wait a minute or two until a window appears)

1.3 A window will appear with the message: "Image Processing Complete" - close this window.

2. DSpace Batch importing

2.1 Open the Command prompt:

- Click the Round Start Button bottom left corner

- Choose Run, then cmd, then OK Button

2.2 Insert the following in the black window: cd c:\dspace\bin (Press Enter)

2.3 After C:\dspace\bin insert the following command: (Copy the text line below and right-click in the Command Prompt window to paste)

dsrun org.dspace.app.itemimport.ItemImport -a -e marius-nel@hotmail.com -c 123456789/326 -s "C:\maps" -m "C:\maps\mapfiles\mapfile.txt"

Press Enter to run the command.

Close the Command Prompt window after the program completes. (When it returns to C:\dspace\bin)

3. View the Large Image in the Browser

3.1 Open Firefox (Click the Firefox tab on the taskbar)

3.2 Refresh the current window

3.3 In the right side bar under “Recent Submissions” - choose Africa.

3.4 Click on the Africa.html link in the grey "Files in This Item Box".

3.5 Click on the image displayed to zoom-in on the map.
- You may also click-and-drag to move the image left and right.
- You may also use the mouse scroll wheel to zoom in and out of the image.
- Use the controls located in the window to inspect the image.

5.3.3. Evaluation results

All of the participants successfully completed the entire process and were able to view a large image in its original resolution. Figure 16 graphically illustrates the results of the evaluation. The total number of participants is plotted along the vertical axis and the outcome of each instruction is plotted along the horizontal axis (represented by Yes or No coloured columns).

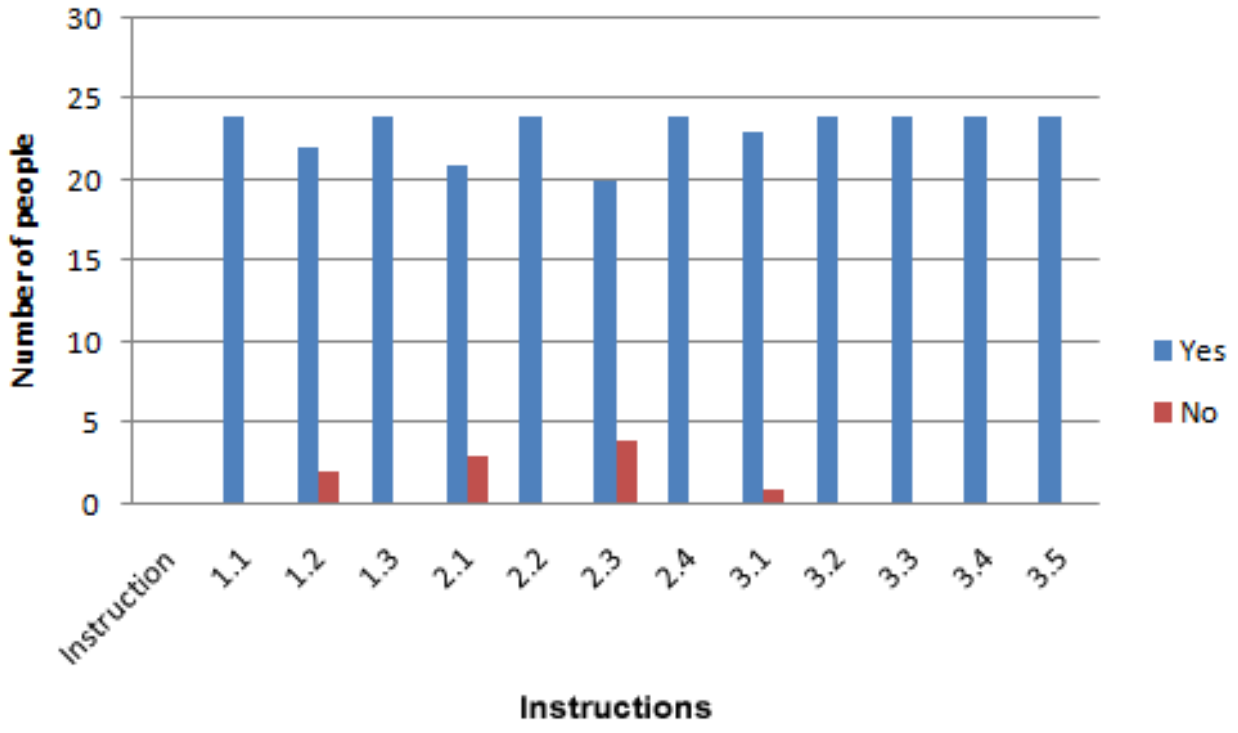

Figure 16 - Results of user-based evaluation of first prototype

Figure 16 indicates that the participants could complete most of the instructions without any problems. There were a total of 10 instances where people failed to complete the instruction successfully of which seven resulted from opening the command prompt and inserting the required command to launch the DSpace batch import facility which was therefore the area where most of the problems occurred. The "Browse For Folder" dialog window step resulted in two mistakes and one person failed to open the Firefox window because of unfamiliarity with this browser.

The time it took each participant to complete the entire process was also recorded and the results are illustrated in Figure 17.

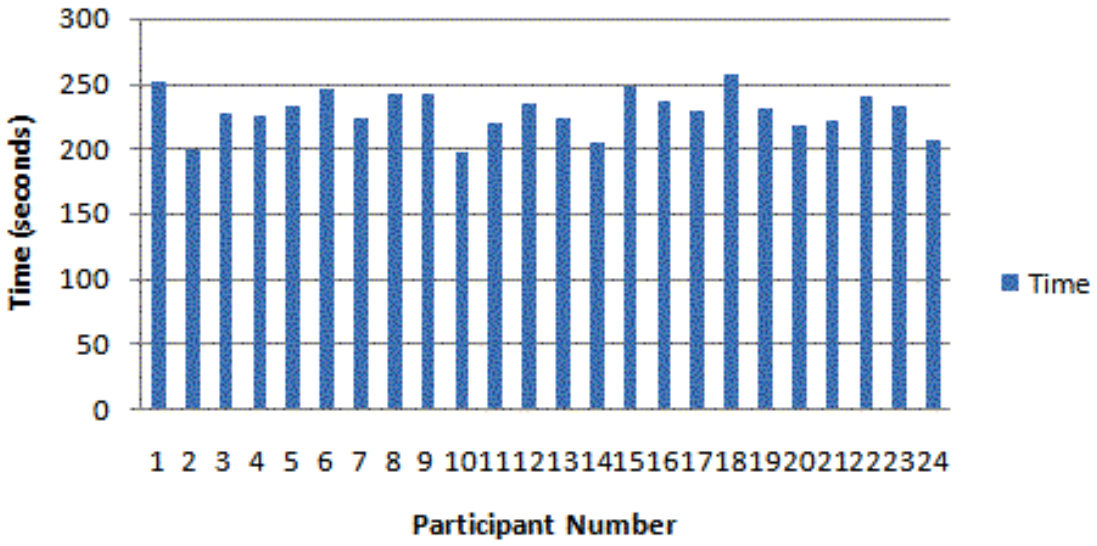

Figure 17 - Recorded timings of first prototype evaluation

The quickest recorded evaluation was 197 seconds and the longest recorded evaluation measured 259 seconds. Therefore a full minute separated the fastest and slowest recorded times with the rest of the recorded times more or less evenly distributed over that period. The reason for the big range in recorded times could be attributed to the variable familiarity of participants with using a command-line utility. 16.7% of the users experienced difficulty in inserting the required commands to launch the batch process. Most of these users mistakenly expected the copy and paste keyboard shortcuts to work in this environment. Furthermore, many participants chose to type some of the commands, which resulted in longer recorded times.

The results of the study highlighted some of the weaknesses of the first prototype. The most evident flaw is the lack of a visual feedback mechanism, such as a progress bar, while the images are being processed. After launching the prototype program, and selecting the folder that contains the image/images to be processed, a lengthy period of seeming inactivity takes place while the program processes the images, with no visual feedback provided to the user to inform them of what is happening. Only after the program has completed the image processing and DSpace preparation procedures, does a window appear informing the user that the images have been processed. This caused 8.3 % of the participants to start clicking around the on the desktop, opening windows and trying to move on to the next step of batch importing even when the instructions clearly stated that the user should wait a minute or two until a window appeared with the message “Image processing completed”.

The viewing client's maximum zoomed-out level was criticised by one of the participants, who suggested that the maximum zoomed-out level should be limited to where the entire image fits in the view-port. The current configuration allows the user to zoom out until the image is 256pixels x 256pixels in size.

It was also noted that none of the participants enabled the full-screen viewing mode. This feature would effectively make the viewing area considerably larger and therefore more effective in inspecting image details. The lack of enabling this feature could most likely be attributed to the fact that the full-screen toggle control is only available via the right-click option menu when inspecting the image.

In the following section presents the design and evaluation of the second prototype system.

5.4. Second high-fidelity prototype

The goals of developing the second high-fidelity prototype included:

1. Improving the usability of the first prototype system,

2. Reducing the number of steps required from the user to complete the process of providing DSpace with large image support

3. Making the prototype less reliant on proprietary software and third party browser plug-ins.

The results and feedback from the user-based evaluation of the first prototype formed the basis for alterations made to improve the prototype system.

5.4.1 Tools employed

The second high fidelity prototype employed the same software tools as the first prototype in terms of image processing and DSpace batch importing. The client-side viewing module however was changed to incorporate *Microsoft's Seadragon Ajax*[23]. Seadragon Ajax is a Deep Zoom viewing library implemented in pure JavaScript. It is freely available under the Microsoft Public License (Ms-PL) [24].

[23] Microsoft Seadragon Ajax (http://www.seadragon.com/developer/ajax/)

[24] Microsoft Public License (http://www.opensource.org/licenses/ms-pl.html)

5.4.2. Client-side viewer

A decision was made to move to a pure JavaScript-based viewing module to ensure that the prototype would be usable on platforms without the need to install third-party browser plug-ins. A JavaScript-based solution also does not require a proprietary development environment such as *Adobe's Flash*.

The default Seadragon Ajax Deep Zoom viewer interface provides zooming and panning controls available via mouse click- and scroll events. Additionally the user-interface also includes buttons to enable zooming in and out of the image, reset the image to original resolution and also a button to enable a full-screen view (see Figure 18).

Figure 18 - Seadragon Ajax interface button elements

A modification to the source code was required to enable the viewer to read image tiles from a flat directory structure to conform to the DSpace requirement that an item and its related files be contained in a flat directory structure.

5.4.3. Image processing

The executable C# Windows Form program developed for the first prototype to batch process a folder of large images was modified to include the following:

1. A Progress Bar to provide the user with a visual feedback mechanism that the program is busy with image processing (see Figure 19).

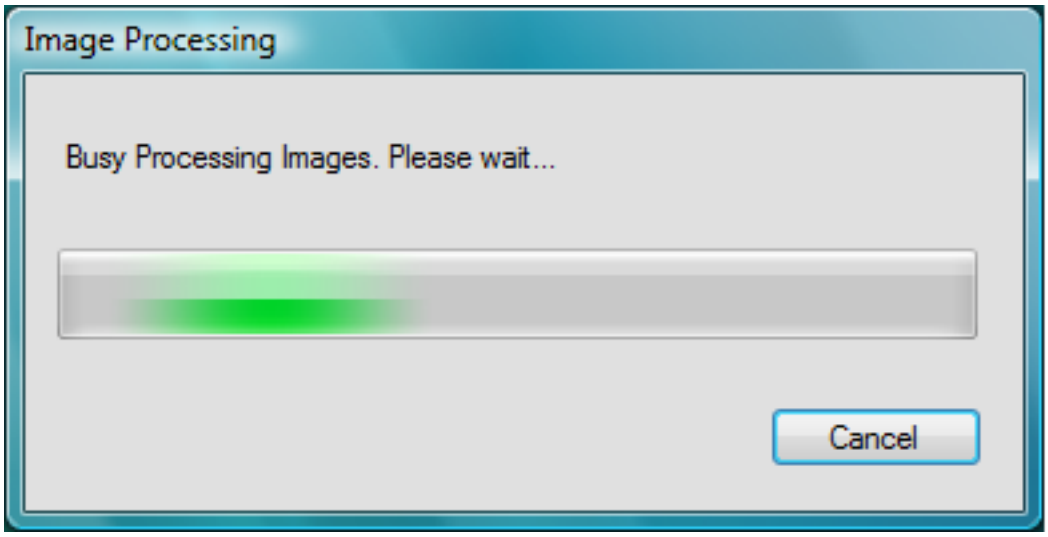

Figure 19 - Image Processing progress bar

2. The generation of an HTML file that includes a link to the new Seadragon Ajax Deep Zoom viewing library JavaScript file.

3. A new form window to capture the relevant DSpace Collection information required for successful DSpace batch importing (see Figure 20).

4. A process to launch the DSpace Batch import command line program after the DSpace Collection information has been captured.

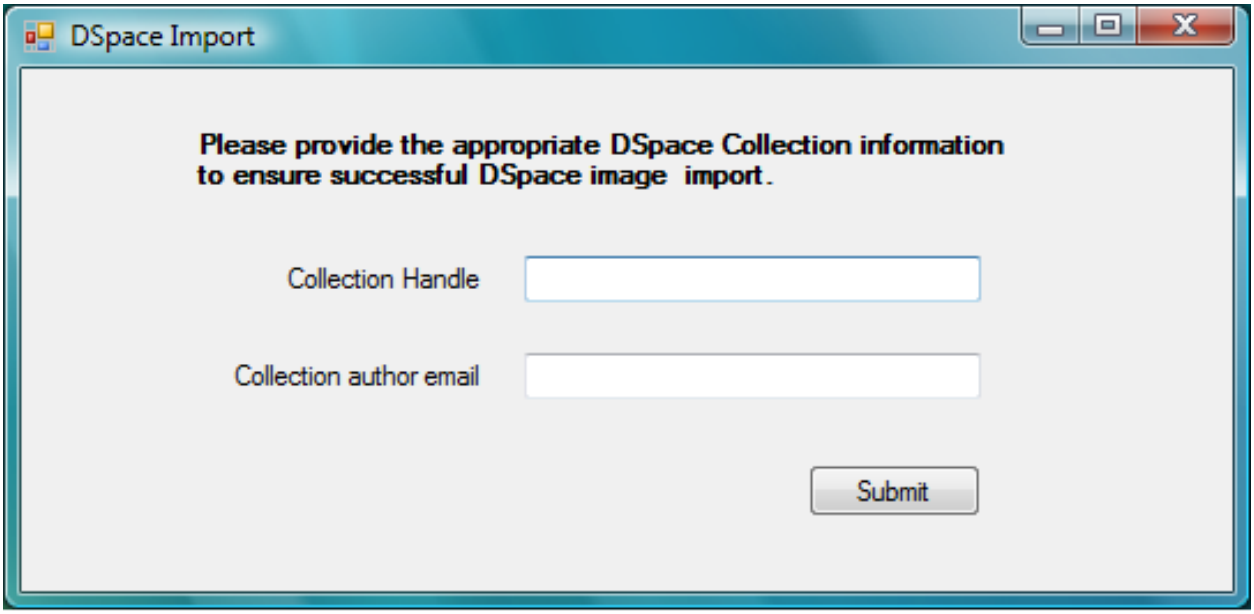

Figure 20 - DSpace Collection information form

5.4.4. DSpace batch importing

In the first prototype the import of all the image assets into DSpace was accomplished by launching DSpace's command-line utility for batch importing of items as a separate process after the images have been processed. In the second prototype this step has been integrated to form part of the image processing process, therefore effectively reducing the number of steps required to complete the process of adding large image support to DSpace and also effectively wrapping all the required processes within one program and eliminating the need to run two separate programs to complete the entire process.

5.4.5 Summary of second prototype

The second high-fidelity prototype included a new JavaScript-based client viewer to remove the prototype system's reliance on proprietary software in terms of a development environment (Adobe's Flash). A JavaScript based viewer also effectively removes the need to install third party browser plug-ins.

The image processing program was improved to include a progress bar to provide visual feedback to the user that the program was busy with image processing. The program also integrated the DSpace batch-importing process by capturing the necessary DSpace collection information in a form window and then launching a shell-process to run DSpace's batch importing script, therefore effectively reducing the number of steps required to complete the entire process.

5.5. Second prototype evaluation

For the second prototype, another user-based evaluation was conducted in the same manner as the first prototype evaluation. In addition, tests were performed to evaluate the system's image processing and batch importing speed. The amount of data transferred while inspecting a high-resolution image was also measured to test whether the prototype system was able to deliver large images in a bandwidth-conscious manner.

5.5.1. Evaluation instruction list

1. Image Processing

1.1 In the open window double click on DeepZoomer.exe

1.2 In the "Browse For Folder" dialogue window - select Computer, then Local Disk (C:), then Maps, then select the OK button.

(A Progress bar will appear to indicate that the program is busy processing images. After the image processing is complete a new form window will appear with two text fields)

1.3 Type (or copy and paste) the following in the first text field labeled "Collection Handle": 123456789/121

1.4 Type (or copy and paste) the following in the first text field labeled "Collection author email": mars@rocknroller.co.za

1.5 Click Submit

(A window with a black background will appear indicating the progress of the DSpace batch import process)

1.6 When the batch import process has completed a window will appear with a message that reads "Image Processing completed" - Close this window

2. View the Large Image in the Browser

2.1 Open Firefox (Click the Firefox tab on the taskbar)

2.2 Refresh the current window

2.3 In the right side bar under "Recent Submissions" - choose Africa.

2.4 Click on the Africa.html link in the grey "Files in This Item Box".

2.5 Click on the image displayed to zoom-in on the map.
- You may also click-and-drag to move the image left and right.
- You may also use the mouse scroll wheel to zoom in and out of the image.
- Use the controls located in the window to inspect the image.

5.5.3. Evaluation results

All of the participants in the user-based evaluation of the second prototype system also successfully completed the entire process of: image processing, DSpace batch import and finally viewing a large image via the client viewer embedded in the HTML file through a browser. Figure 21 graphically illustrates the results of the second evaluation. The total number of participants is plotted along the vertical axis and the outcome of each instruction is plotted on the horizontal axis (represented by Yes or No coloured columns).

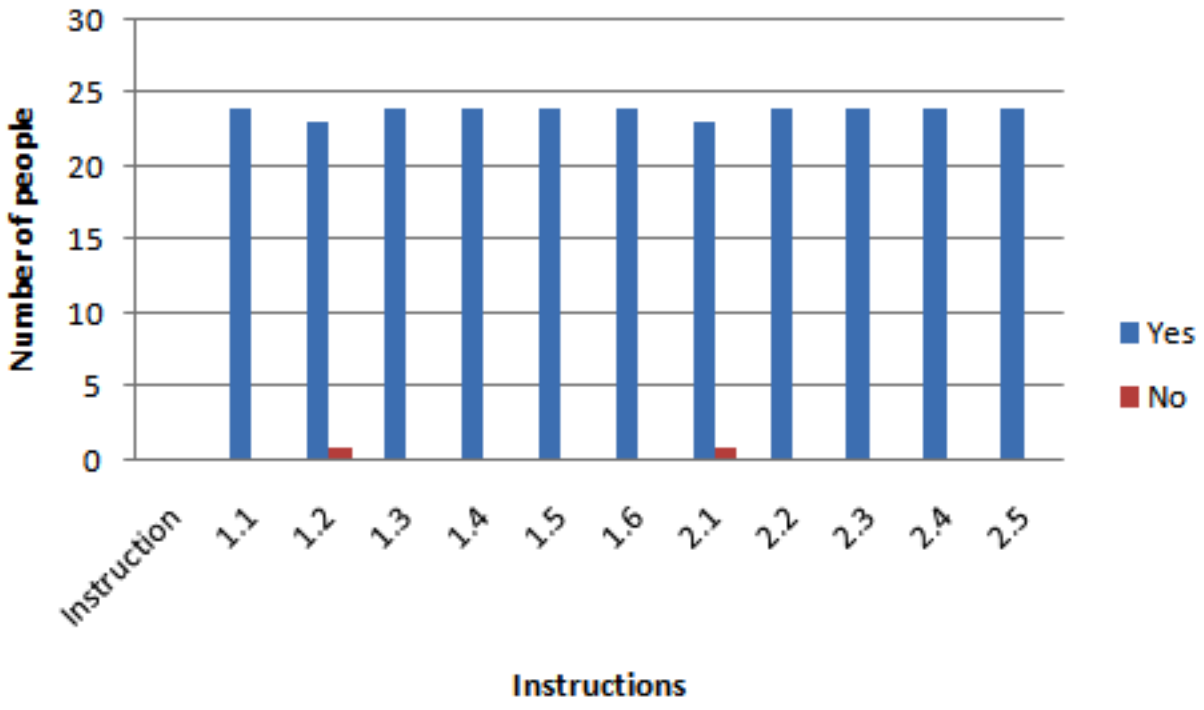

Figure 21 - Results of user-based evaluation of second prototype

Figure 21 indicates that two errors occurred while the participants completed the evaluation study. At instruction number 1.2, one person chose the wrong directory in the "Browse for Folder" dialogue window and another error occurred at instruction number 2.1, where the person failed to locate and open the Firefox tab in the taskbar. The number of errors was significantly lower in the second prototype evaluation, which may be the result of reducing the number of steps required from the user to complete the process of providing DSpace with large image support.

The time to complete the process was also measured and the results of these measurements are illustrated in figure 22.

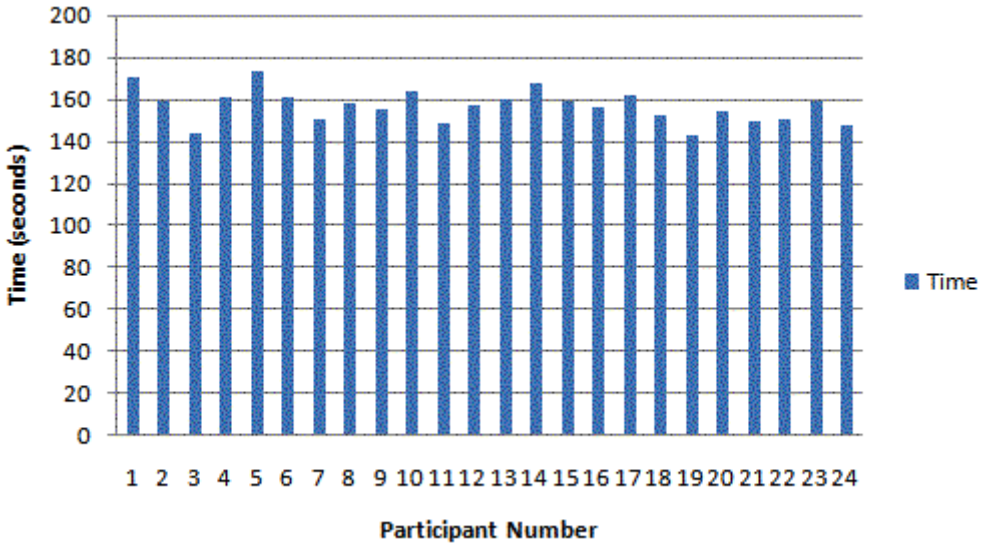

Figure 22 - Recorded timings of second prototype evaluation

The second prototype evaluation's recorded times revealed a significant reduction in the time it took the participants to complete all the steps in the evaluation instruction list. The recorded times ranged from 143 seconds to 173 seconds. The fastest time of the second evaluation was 54 seconds faster than the fastest recorded time from first prototype evaluation while the slowest time of the second prototype evaluation was still 24 seconds faster than the quickest time of the first prototype evaluation. The difference between the fastest and slowest times was 30 seconds, with the rest of the results more or less evenly distributed. These reductions could be attributed to the fact that the DSpace import process has been simplified and has also been integrated to form part of the image processing application. The total number of steps required to complete the process has therefore been reduced from 19 user interactions to 13 user interactions.

The results show that almost all the tasks were completed with no errors. Only two of the tasks produced single errors that were easily recoverable. One participant chose the wrong drive in the "Browse for folder" task but managed to quickly recover from this mistake. One participant mistakenly closed the running instance of the Firefox browser window that displayed the DSpace collection which then had to be re-opened. These mistakes did not indicate any failures in terms of the prototype system's design.

Some of the positive comments included "That was easy", "loved the full-screen view" and "smooth zooming effect". The zooming functionality did however get some negative comments. One person remarked that it was "a bit slow" and another commented that the blurring between the zooming levels was "slightly disorientating".

5.6. Performance testing

In this section the prototype's image processing performance and bandwidth efficiency in terms of large image access is evaluated.

5.6.1. Image processing

To measure the prototype's image processing speed, an 81 mega-pixel (9690 pixels wide by 8373 pixels high) image was saved in three common file formats (PNG, JPEG & TIFF) to use as a test case. The time to produce the image assets for DSpace import was measured for each of the three images. The image processing times of two commercial large image applications, Zoomify[25] and Microsoft's DeepZoom Composer[26], are also measured using the three test images. The results of these tests are displayed in the table below. The testing computer's specifications are listed in Appendix C.

[25] Zoomify (http://www.zoomify.com/)

[26] Microsoft Deep Zoom Composer (http://www.microsoft.com/downloads/details.aspx?FamilyID=457B17B7-52BF-4BDA-87A3-FA8A4673F8BF&displaylang=en)

PROTOTYPE IMAGE PROCESSING						
Image		Output data				
Format	Size	Import Time	Processing Time	Total Time	Files	Size
TIFF	232 MB	0	1 min 26 s	1 min 26 s	1770	22.3 MB
PNG	150 MB	0	1 min 29 s	1 min 29 s	1770	22.4 MB
JPEG	16.6 MB	0	1 min 25 s	1 min 25 s	1770	21.7 MB

ZOOMIFY IMAGE PROCESSING						
Image		Output data				
Format	Size	Import Time	Processing Time	Total Time	Files	Size
TIFF	232 MB	13 s	0 min 37 s	0 min 50 s	1709	31.6 MB
PNG	150 MB	12 s	0 min 36 s	0 min 48 s	1709	31.6 MB
JPEG	16.6 MB	10 s	0 min 35 s	0 min 45 s	1709	27.6 MB

MICROSOFT DEEPZOOM COMPOSER IMAGE PROCESSING						
Image		Output data				
Format	Size	Import Time	Processing Time	Total Time	Files	Size
TIFF	232 MB	1 min 14 s	0 min 38 s	1 min 52 s	1717	23.3 MB
PNG	150 MB	1 min 16 s	0 min 38 s	1 min 54 s	1717	23.3 MB
JPEG	16.6 MB	1 min 10 s	0 min 38 s	1 min 48 s	1716	22.4 MB

Table 1- Image Processing Test Results

The results from the three tests show that the prototype system's image processing performance was significantly slower than Zoomify and Deep Zoom Composer. However, the two commercial solutions require extra time to open or import an image for processing. Deep Zoom Composer is in reality the slowest of the three systems if you take the import time into consideration. Zoomify's total image processing speed is the fastest of the three, although the total size of its output files are the biggest of the three. The prototype's output produces the biggest number of files but with smallest size in total. This may be due to the fact that the prototype relies on a predefined compression ratio when processing images. The two commercial applications allow a user to set the compression ratio before processing and a setting of 80% percent JPEG compression was used in both in this test.

5.6.2 Bandwidth efficiency

A Web-based experiment was conducted to measure the bandwidth efficiency of the prototype implementation. The images used for testing purposes were high-resolution scans of historical hand-drawn maps of the African continent. These images formed part of University of Cape Town's (UCT) Atlas of Mutual Heritage collection and were obtained from UCT's African Studies Library. A total of six images of various sizes (resolution) were selected. Half of these images were compressed with standard high quality JPEG compression, while the other half employed 24-bit PNG compression.

The amount of data transferred while inspecting large images for a period of 5 minutes was measured to identify whether the prototype solution succeeded in presenting large images in a bandwidth-conscious manner.

The test also measured bandwidth efficiency at different rates of Internet connectivity. The first test was conducted with a desktop Personal Computer (PC) connected via a wireless network to a DSL modem with a connection speed of 34 KB per second. The second test was conducted with a 3G cellular modem connected to the same PC with a connectivity speed of 230 KB per second. Finally a third test was conducted with a mobile device that was wirelessly connected to the DSL modem with a connectivity speed of 34 KB per second, to test bandwidth efficiency in terms of mobile client access.

There was no limit imposed on the number user interactions (mouse-clicks and scroll-wheel) during the inspection of the images. Higher-resolution image data consist of more image tiles than lower-resolution images and therefore require a higher number of user interactions to access the image tiles at the original resolution. During the test, all the images were inspected at various resolution levels, including the original resolution of the image.

The table below (Table 2) documents the results of these tests.

						Bandwidth Usage (Megabytes)			
Client	**Resolution (Megapixel)**	**Size (MB)**	**File Type**	**Speed / Second**	**Period**	**Down**	**Up**	**Total**	**Percentage of original**
PC	71	11.2	JPEG	34 KB	5 min	4.5	0.5	5	44.64
PC	138	26	JPEG	34 KB	5 min	8.4	0.6	9	34.61
PC	138	36.6	JPEG	34 KB	5 min	8.6	0.6	9.2	25.13
PC	57	101.8	PNG	34 KB	5 min	11.7	0.6	12.3	12.1
PC	83	144.7	PNG	34 KB	5 min	12.3	0.7	13	8.9
PC	83	154	PNG	34 KB	5 min	12.6	0.7	13.3	8.6
Total		**474.3**				**58.1**	**3.7**	**61.8**	**13.03**
PC	71	11.2	JPEG	230 KB	5 min	8.5	0.6	9.1	81.25
PC	138	26	JPEG	230 KB	5 min	10.1	0.9	11	42.31
PC	138	36.6	JPEG	230 KB	5 min	10.5	1	11.5	31.42
PC	57	101.8	PNG	230 KB	5 min	22	2.2	24.2	23.77
PC	83	144.7	PNG	230 KB	5 min	24	2.3	26.3	18.18
PC	83	154	PNG	230 KB	5 min	24.6	2.3	26.9	17.47
Total		**474.3**				**99.7**	**9.3**	**109**	**22.98**
iPod	71	11.2	JPEG	34 KB	5 min	4.1	0.3	4.4	39.28
iPod	138	26	JPEG	34 KB	5 min	7	0.4	7.4	28.46
iPod	138	36.6	JPEG	34 KB	5 min	7.2	0.4	7.6	20.77
iPod	57	101.8	PNG	34 KB	5 min	7.9	0.4	8.3	8.2
iPod	83	144.7	PNG	34 KB	5 min	8.5	0.4	8.9	6.2
iPod	83	154	PNG	34 KB	5 min	8.6	0.4	9	5.84
Total		**474.3**				**43.3**	**2.3**	**45.6**	**9.61**

Table 2 – Prototype bandwidth usage

The table shows that the total amount of data transferred during a 5-minute period of inspection was less than the original image file size for all the test cases. In the first test, with a connectivity speed of 34 KB per second, the total amount of data transferred for all 6 images accounted for 13% of the total image sizes. The second test results show that 23% of the combined image sizes were transferred at a connectivity speed of 230 KB per second and the third test revealed that only 9.6% of the total combined image sizes were transferred while inspecting the images with a mobile device at a speed of 34 KB per second.

The higher connectivity speed of 230 KB per second compared to 34 KB per second resulted in an increase of 47. 2 MB of data transferred while the smaller screen size of the mobile device may account for the decrease of 16.2 MB of data transferred during the mobile device test.

5.7. Concluding remarks

The user-based evaluation studies revealed that the prototype system successfully provided large image support to the DSpace platform. All the participants successfully completed the three-part process of processing a folder of large images in order to prepare it for the DSpace batch import, importing the images into a DSpace collection and inspecting and viewing the large image with the controls that the prototype's client viewing module provides.

The prototype solution's image processing performance was also measured and compared to two commercial large image applications' performance. It was found that the prototype's image processing was the second fasted of the three solutions if the time it takes to import or open an image was taken into consideration.

The prototype's bandwidth consumption was also measured while image data was served via the Web to the client viewer to measure its bandwidth efficiency in terms of image access. It was found that only a fraction of the total image data was transferred during image inspection. This proved that the prototype system was effective in reducing the amount of data transferred in the process of inspecting a high-resolution image.

6. Conclusion

This research aimed at finding a solution that will provide large image support to a popular DR platform. Many organizations are building digital repositories in an effort to preserve and provide access to their digital materials. Multimedia, especially images and video are important elements of these libraries. To meet the needs of preservation, images are captured in an uncompressed or lossless format to preserve as much of the original detail as possible. This process results in high-resolution images stored in large files, which pose a problem in terms of providing Web access to these high-resolution images.

The literature review revealed several methods that effectively overcome the problems of accessing large images over the Web. Most of these solutions are based around the idea of a *zooming user interface* (ZUI). ZUIs provide an infinite canvas that can be accesses through zooming and panning controls. The images accessed through a ZUI are constructed from multiple derivative images or an image pyramid where the original large image is processed to produce 'tiles' of the image at different scales. Each pan and zoom action causes a request for a small number of additional tiles – those that form part of the image panned to, at the level of zoom desired. These additional tiles are then streamed on-demand, to the viewer. No tiles are therefore delivered unless required for the current display, which helps to limit the amount of data transferred while inspecting the image.

The literature review further revealed two image preservation formats, TIFF and JPEG 2000 that provide the ability to support multiple derivatives of the original image within one file. It can for example, provide multiple reduced resolution versions of the original, or a high-resolution, high quality view of a portion of the image with a region of interest without the need to pre-generate these image files. Currently however, there exist no native browser software support for these image formats and therefore these features cannot be utilized without relying on a browser plug-in or a server-side method that translates the requested image file into a browser-compatible format.

The integration of a large image support system into a popular DR has seldom been investigated. This research therefore pursued the development of a solution based on utilizing existing open source ZUI technologies to provide effective large image support to DSpace, a popular DR platform.

To gather user feedback data with regards to the usability and efficacy of current large image support solutions that have been implemented within DRs and also commercial solutions that exist beyond the scope of digital repositories, a survey was conducted that focused on four different large image solutions. Two of these solutions represented commercial applications while the other two represented custom large image solutions that have been implemented within

DR systems. The results of this survey were used to inform the design and development of the prototype large image solution.

The development of the prototype adhered to the principles of user-centered design (UCD), where evaluation by users form part and parcel of the prototype development, where the users' suggestions for improvements are implemented in the iterative process of prototyping. Two user-based evaluation studies were conducted as part of this process to test the usability of prototype system. The problems observed and the feedback gathered from the first user-based evaluation was analysed to make design decisions to in the development of the second prototype implementation.

Finally, two experiments were also conducted to measure the image processing performance and bandwidth efficiency of the final prototype system.

6.1. Research questions

The studies and experiments conducted aimed to answer the following questions.

Are there solutions available that deal effectively with the problems of delivering large images over the Internet?

From the literature review and large image support survey conducted, it was found that several solutions rely on ZUI technologies to successfully provide access to high-resolution images on the Web.

Do digital repositories provide support for delivering large images over the Internet?

Although some organizations have implemented custom-built large image solutions within their digital repositories, none of the popular open source digital repository systems provide native support for high-resolution images.

Is it possible to incorporate an effective large solution within a popular Digital Repository?

The user-based evaluation revealed that the prototype system successfully provided a usable solution that effectively provides large image support to DSpace, a popular open source digital repository. This included the following tasks: image processing, DSpace batch import and finally viewing a large image via the client viewer embedded in the HTML file.

6.2. Contributions

This research focused on providing a usable, effective large image support solution for a popular digital repository software platform. The literature survey revealed two possible approaches to providing access to high-resolution images over the Internet: one method relies on the use of ZUIs and multiple derive image pyramid technology, while the other suggested the possibility of taking advantage of some image file formats' capacity to provide multiple resolution instances of the same image contained within one file, without the need to pre-generate extra image files.

The literature review revealed that the integration of an effective large image solution within a popular digital repository platform has not been pursued. This research proves that by utilising and extending currently available open source large image processing and displaying software, large image support could be provided to the DSpace platform with minimal or no modification to the DSpace source code. The proposed solution is based on providing access to a high-resolution image through the use of a ZUI embedded in an HTML file and pre-generated multiple derive images that is imported through a DSpace batch import facility in a seamless easy to use process.

This work however, does not limit any future work of developing a solution that takes advantage of TIFF and/or JPEG 2000's capacity to provide multi-resolution views of an image from the same file. Furthermore, as acceptance and use of the JPEG 2000 file format grows, further research into this aspect is encouraged.

6.3. Future work

There are a number of areas where this research could be extended. The following discuss some of these areas.

6.3.1. JPEG 2000

The growth of JPEG 2000 as an acceptable preservation format coupled with an impressive feature set makes it an important area of future research. Possible future work could involve extending the client-viewer to incorporate support for the JPEG 2000 format. More specifically to take advantage of JPEG 2000's multiple resolutions and region of interest extraction to eliminate the need to store pre-generated tiles for all resolution levels and therefore reduce storage costs.

6.3.2. Batch import

Possible future work would involve extending the batch import facility to enable batch ingestion via the Internet. At this point, the prototype system is configured to accept local batch imports, which requires local access to the server that host the DSpace installation. Batch ingestion via the

Internet would make it possible to submit large image collection data from anywhere in the world.

6.3.3. Integration with other DR platforms

Further investigation in terms of the possibility of integrating this solution with other DR platforms such as Fedora and EPrints is another area of possible future work. The prototype's image processing interface could for instance, be modified to offer the user the choice of which platform to use and as the result of that choice initiated DR specific processes that will result in effective large image support for that particular platform.

6.3.4. Operating System compatibility

Currently the solution is built to operate on a Windows-based operating system. The prototype application could also be ported to Linux-based operating systems by incorporating open-source image processing tools and programming languages that will run on Linux operating systems.

References

Abilene Library Consortium (n.d). West Texas Digital Archives. Retrieved Jan 03, 2010, from http://wtda.alc.org/

Adobe (2008). Adobe Flash Player 10 Security white paper, Retrieved from http://www.adobe.com/content/dam/Adobe/en/devnet/flashplayer/pdfs/flash_player_10_security.pdf

Allinson J., François S., & Lewis S. (2008). SWORD: Simple Web-service Offering Repository Deposit. *Ariadne, 54.* Retrieved from http://www.ariadne.ac.uk/issue54/allinson-et-al/

Association of Research Libraries (ALR), (2009). The Research Library's Role in Digital Repository Services. Retrieved from http://www.arl.org/bm~doc/repository-services-report.pdf

Barry, R. E. (2002). Managing Distinctions: Enterprise Information, Document, Records, Knowledge and Content Management. *Records and Information Management Review, 18:2.* Retrieved from http://www.mybestdocs.com/barry-r-im-rm-distinctions.htm.

Bederson B., Hollan, J. D., Perlin, K., Meyer, J., Bacon, D., & Furnas, G. W. (1996). Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. *Journal of Visual Languages and Computing, 7, 3-31.*

Borghoff, U., Rödig, P., Scheffczyk, J. & Schmitz, L. (2006, October). Long-Term Preservation of Digital Documents: Principles and Practices. Springer, 1.

Buonora, P. & Liberati, F. (2008, July/August). A Format for Digital Preservation of Images A Study on JPEG 2000 File Robustness". *D-Lib Magazine, 14, 7/8.*

Buckley, R. (2008, February). JPEG 2000 - A Practical Digital Preservation Standard? *DPC Technology Watch Series Report 08-01.*

Chute R., & Van de Sompel H. (2008, September/October). Introducing Djatoka A Reuse Friendly, Open Source JPEG 2000 Image Server, Los Alamos National Laboratory, *Research Library.D-Lib Magazine, 14, 9/10.*

Donkin, S. (2001). Digital images forever: implementing an imaging system in a cultural institution. Computing Arts: Digital Resources for Research in the Humanities. University of Sydney. Retrieved from http://ses.library.usyd.edu.au/handle/2123/6211

Dumas J.S. & Redish J.C. (1999) A Practical Guide to Usability Testing (revised ed). Exeter UK: Intellect Ltd.

Federal Agencies Digitization Initiative (FADGI) - Still Image Working Group. (2009). Technical Guidelines for Digitizing Cultural Heritage Materials: Creation of Raster Image Master Files, Technical Guidelines. Retrieved from http://www.digitizationguidelines.gov/guidelines/digitize-technical.html

Freeman, M. (2008) The Complete Guide to Digital Photography (4th ed). Asheville, North Carolina: Lark Books.

Gillesse, R., Rog, J. & Verheusen, A. (2008). Alternative File Formats for Storing Masters. National Library of the Netherlands Research & Development Department. Retrieved from http://www.kb.nl/hrd/dd/dd_links_en_publicaties/publicaties/Alternative_File_Formats_for_Storing_Masters_2_1.pdf

Hodge G. & Anderson N. (2007). Formats for digital preservation: A review of alternatives and issues. *Information Services & Use, 27, 45–63.*

Jones, M. & Marsden, G. (2006). *Mobile Interaction Design*. Chichester, West Sussex: John Wiley & Sons.

Jones R. (2004). The Tapir: Adding E-Theses Functionality to DSpace. *Ariadne, 41*. Retrieved from http://www.ariadne.ac.uk/issue41/jones/intro.html

Körber, N. & Suleman, H. (2008). Usability of Digital Repository Software: A Study of DSpace Installation and Configuration. Proceedings of ICADL '08. Retrieved from http://pubs.cs.uct.ac.za/archive/00000492/01/2008_06_15_icadl_submission.pdf

Kahn R. & Wilensky R. (1995) A Framework for Distributed Digital Object Services. *International Journal on Digital Libraries*, 6(2), 115–123. Retrieved from http://www.doi.org/topics/2006_05_02_Kahn_Framework.pdf

Kulovits, H., Rauber A., Kugler, A., Brantl, M., Beinert T. & Schoger, A. (2009, November/December). From TIFF to JPEG 2000? - Preservation Planning at the Bavarian State Library Using a Collection of Digitized 16th Century Printings. *D-Lib Magazine, 15, 11/12.*

Liogkas N. & Tungare M. (2002). Zoomable User Interfaces – Literature Review. Unpublished, Retrieved from http://www.leogas.net/documents/ZUI_Literature_Review.pdf

Lowe, D. & Bennett, M. J. (2009). A Status Report on JPEG 2000 Implementation for Still Images: The UConn Survey. *UConn Libraries Published Works.* Paper 19. Retrieved from http://digitalcommons.uconn.edu/libr_pubs/19

Lagoze C., Payette S., Shin E. & Wilper C. (2005). Fedora - An Architecture for Complex Objects and their Relationships. Computing and Information Science, Cornell University, Retrieved from http://arxiv.org/pdf/cs.DL/0501012

Lyman, P. & Varian, H.R. (2003). How much information? 2003. School of Information Management and Systems, University of California at Berkeley. Retrieved from http://www2.sims.berkeley.edu/research/projects/how-much-info-2003

Martin R. (2003) ePrints UK: Developing a national e-prints archive. *Ariadne, 35*. Retrieved from http://www.ariadne.ac.uk/issue35/martin/

McKay D. (2007). Institutional Repositories and Their 'Other' Users: Usability Beyond Authors. *Ariadne, 52.* Retrieved from http://www.ariadne.ac.uk/issue52/mckay/

Microsoft (2010). Seadragon Ajax. Retrieved Jan 03, 2010, from http://seadragon.com/developer/ajax/

Nixon W. (2003). DAEDALUS: Initial experiences with EPrints and DSpace at the University of Glasgow, *Ariadne, 37.* Retrieved from http://www.ariadne.ac.uk/issue37/nixon/intro.html

Norman, D. (1988). The Design of Everyday Things. New York: Basic Book.

Northwestern University Library (n.d). Africana Collection, Retrieved Jan 03, 2010, from http://ansel.library.northwestern.edu/ImageServer/flash.jsp?filename=/dimages/public/images/inu-afrps/inu-afrps-af0005-78.jp2&title=inu-afrps-af0005-78.jp2

OpenDOAR. (2011). University of Nottingham, UK. OpenDOAR.org.

Payette, S. & Lagoze, C. (1998). Flexible and Extensible Digital Object and Repository Architecture (FEDORA). Proceedings from Second European Conference on Research and Advanced Technology for Digital Libraries. Heraklion, Crete.

Peneva J., Ivanov S., Andonov F. & Dokev N. (2009) Digital objects – storage, delivery and reuse. *International Journal Information Technologies & Knowledge, 3, 61-70.* Retrieved from http://www.foibg.com/ibs_isc/ibs-14/ibs-14-p08.pdf

Phillips S., Green C., Maslov A., Mikeal A., & Leggett J. (2007, November/December). Manakin - A New Face for DSpace, *D-Lib Magazine*, *13, 11/12.*

Pinfield S., Gardner M., & MacColl J. (2002, March/April). Setting Up an Institutional e-Print Archive. *Ariadne,* 31. Retrieved from http://www.ariadne.ac.uk/issue31/eprint-archives/

Saleh I., Adly N. and Nagi M. (2005). DAR: A Digital Assets Repository for Library Collections. Research and Advanced Technology for Digital Libraries Lecture Notes in Computer Science, 3652, 116-127.

Sharp, H., Rogers, Y., & Preece, J. (2007). Interaction Design: Beyond Human-Computer Interaction (2nd ed.). Chichester, West Sussex: John Wiley & Sons.

Smith, A. (2007). Zoomify Image. Cornell University Library, Retrieved from http://ecommons.library.cornell.edu/handle/1813/5410

Smith, I. (2008). Digital preservation through using DSpace open source institutional repository software: a case study. The University of Pretoria. Unpublished. Retrieved from www.ais.up.ac.za/digi/docs/smith_paper.pdf

Staples, T. & Wayland, R. (2000). Virginia Dons FEDORA: A Prototype for a Digital Object Repository. D-Lib Magazine, 6, 7/8.

Tedd, L. & Large, A. (2005). Digital Libraries - Principals and Practice in a Global Environment. Munich: K.G. Saur.

Tansley R., Bass M., Branschofsky M., Carpenter G., McClellan G., & Stuve D. (2005). DSpace System Documentation, Retrieved from http://dspace.org/technology/system-docs/

Tansley, R., Bass, M. & Smith, M. (2003). DSpace as an open archival information system: Current status and future directions. In: Koch, T., Solvberg, I.T. (eds.): Research and Advanced Technology for Digital Libraries. *Lecture Notes in Computer Science, 2769, 446-460.*

Van de Sompel, H., Bekaert, J., Liu, X., Balakireva, & L., Schwander, T. (2005). aDORe. A modular standards-based Digital Object repository. *The Computer Journal, 48(5), 514-535*. Retrieved from http://comjnl.oxfordjournals.org/cgi/content/abstract/48/5/514

Van de Sompel, H., Chute, R. & Hochstenbach, P. (2008). The aDORe Federation Architecture. International Journal on Digital Libraries, 9, 2, 83-100.

Van House, N., Butler, M., Ogle, V., & Schiff, L. (1996). User-Centered Iterative Design for Digital Libraries, The Cypress Experience. School of Information Management and Systems, University of California. D-Lib Magazine, 2, 2. Retrieved from http://www.dlib.org/dlib/february96/02vanhouse.html

Vassiliadis, B., Stefani, A., Drossos L. & Ioannou, K. (2005). Knowledge Discovery in Multimedia Repositories: the role of metadata. Proceedings of 7th WSEAS Int. Conf. on Mathematical Methods and Computational Techniques in Electrical Engineering, Sofia.

Wiggins, R., Davidson, H., Harnsberger, H., Lauman, J. & Goede, P. (2001, May). Image File Formats: Past, Present, and Future. RadioGraphics, 21, 789-798. Retrieved from http://radiographics.rsna.org/content/21/3/789.full.pdf+html

Zierau, E., Jensen C. (2010). Preservation Of Digitised Books In A Library Context. The Royal Library of Denmark Dep. of Digital Preservation Retrieved from http://www.ifs.tuwien.ac.at/dp/ipres2010)/papers/zierau-18.pdf

Zoomify Inc. (2003). Zoomify Technology & Products White Paper. Retrieved Nov 03, 2009, from http://www.zoomify.com

@mire (2008). Add-on Modules for DSpace. Retrieved Feb 03, 2011, from http://atmire.com/dspace_products.php

Appendix A

1. Survey Results

1.1 Survey results for the Large Image Support of the "West Texas Digital Archives" Website (Institutional Repository):

Field Summary for 001:		
Were you able to zoom-in on the image from "West Texas Digital Archives" Website?		
Answer	**Count**	**Percentage**
No answer	0	0
Yes (Y)	24	100.00%
No (N)	0	0

Field Summary for 002:		
Did you need to install additional software to be able to zoom in on the image?		
Answer	**Count**	**Percentage**
No answer	0	0
Yes (Y)	0	0
No (N)	24	100.00%

Field Summary for 003:		
Rate how successful you were in zooming-in on the image:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	1	4.17%
Below average (2)	0	0
Average (3)	6	25.00%
Good (4)	12	50.00%
Excellent (5)	5	20.83%

Field Summary for 004:		
Rate the control mechanisms that enables you to zoom-in on the image:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	3	12.50%
Below average (2)	2	8.33%
Average (3)	12	50.00%
Good (4)	6	25.00%
Excellent (5)	1	4.17%

Field Summary for 005:		
Rate the feedback mechanisms provided by the zooming function:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	4	16.67%
Below average (2)	7	29.17%
Average (3)	9	37.50%
Good (4)	3	12.50%
Excellent (5)	1	4.17%

Field Summary for 006:		
Rate the response speed of the zooming functionality?		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	9	37.50%
Below average (2)	6	25.00%
Average (3)	7	29.17%
Good (4)	2	8.33%
Excellent (5)	0	0

Field Summary for 007:		
Rate the effectiveness of the "West Texas Digital Archives" Large Image Support		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	5	20.83%
Below average (2)	4	16.67%
Average (3)	9	37.50%
Good (4)	6	25.00%
Excellent (5)	0	0

West Texas Digital Archives" Website Feedback/Comments Summary:

2	maximize function is great. there was no clutter of navigation buttons etc. - great! would be nice to have the option to open a small P-in-P which gives you real-time feedback as to where you are in the actual footage...
5	The image took awhile to load, might be my internet connection.
8	no loading image function, poor download speed could be attributed to slow connection my side
14	Images took ages to load (on this connection), so was a little frustrating as I wanted it to be instant! Think it would be better if the user was prevented from dragging once the image had reached the edge of the frame.
15	Image is no good quality off the zoom
18	Connection is slow, so image takes longer to load...
19	Marius, I am not sure what "Rate the effectiveness of the "West Texas Digital Archives" Large Image Support" that means. So I answered poor, it could be that I just don't understand the question properly so it may skew the results. x
21	UCT's poor Internet connection doesn't provide for a reasonable test.
24	I think that there is a major usability issue due to two factors: The delay of loading without any feedback to the user (Just a white screen), and the zoom functionality being a button rather than a mouse control.

28	Image seems to reload each time I press the zoom button which means I have to wait. There is no indication of what is going on - whether it is still loading parts of the image or if I have just reached the edge of the image. Had no control of how much to zoom in with each click. When zooming in I can get lost with no reference to the earlier images. A mini-map would be nice perhaps. I can\'t choose which section to zoom in on, it just magnifies the centre.

1.2 Survey results for the Large Image Support of the "Zoomify" Website (Non-Institutional Repository based solution):

Field Summary for 001:		
Were you able to zoom in on the image from the Zoomify Website?		
Answer	**Count**	**Percentage**
No answer	0	0
Yes (Y)	23	95.83%
No (N)	1	4.17%

Field Summary for 002:		
Did you need to install additional software to be able to zoom in on the image?		
Answer	**Count**	**Percentage**
No answer	0	0
Yes (Y)	2	8.33%
No (N)	22	91.67%

Field Summary for 003:		
Rate how successful you were in zooming-in on the image:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	1	4.17%
Below average (2)	1	4.17%
Average (3)	0	0

Good (4)	11	45.83%
Excellent (5)	11	45.83%

Field Summary for 004:		
Rate the control mechanisms that enables you to zoom-in on the image:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	2	8.33%
Below average (2)	0	0
Average (3)	2	8.33%
Good (4)	13	54.17%
Excellent (5)	7	29.17%

Field Summary for 005:		
Rate the feedback mechanisms provided by the zooming function:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	3	12.50%
Below average (2)	1	4.17%
Average (3)	5	20.83%
Good (4)	7	29.17%
Excellent (5)	8	33.33%

Field Summary for 006:		
Rate the response speed of the zooming functionality?		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	2	8.33%
Below average (2)	0	0

Average (3)	1	4.17%
Good (4)	11	45.83%
Excellent (5)	10	41.67%

Field Summary for 007:		
Rate the effectiveness of Zoomify's Large Image Support		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	1	4.17%
Below average (2)	1	4.17%
Average (3)	2	8.33%
Good (4)	10	41.67%
Excellent (5)	10	41.67%

Zoomify Website Feedback/Comments Summary:

2	over-all look and feel is functional and easy on the eye. no maximize function was a bit frustrating.
5	Good zoom function, nice interface
8	no loading image function
14	More instant in comparison to the previous execution. Controls more intuitive. Like the blurry to clear aspect - much better than blank
18	Mostly made use of clicking on the image to zoom in, but slider also useful
19	Marius: I didn't understand what this question meant "Rate the feedback mechanisms provided by the zooming function:" so I marked as poor, which may skew yr results.
21	I wasn't successful, so the additional questions weren't helpful
25	Use the mouse roller I could zoom out but not in. I had to use the manual controls below the image to zoom in.

1.3 Survey results for The Northwestern University Library Website's (Institutional Repository):

Field Summary for 001:		
Were you able to zoom-in on the image from "Northwestern University Library" Website?		
Answer	**Count**	**Percentage**
No answer	0	0
Yes (Y)	24	100.00%
No (N)	0	0

Field Summary for 002:		
Did you need to install additional software to be able to zoom in on the image?		
Answer	**Count**	**Percentage**
No answer	0	0
Yes (Y)	0	0
No (N)	24	100.00%

Field Summary for 003:		
Rate how successful you were in zooming-in on the image:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	4	16.67%
Below average (2)	1	4.17%
Average (3)	5	20.83%
Good (4)	10	41.67%
Excellent (5)	4	16.67%

Field Summary for 004:		
Rate the control mechanisms that enables you to zoom-in on the image:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	7	29.17%
Below average (2)	5	20.83%
Average (3)	6	25.00%

Field Summary for 005:		
Rate the feedback mechanisms provided by the zooming function:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	8	33.33%
Below average (2)	6	25.00%
Average (3)	7	29.17%
Good (4)	3	12.50%
Excellent (5)	0	0

Field Summary for 006:		
Rate the response speed of the zooming functionality?		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	3	12.50%
Below average (2)	3	12.50%
Average (3)	9	37.50%
Good (4)	7	29.17%
Excellent (5)	2	8.33%

Field Summary for 007:		
Rate the effectiveness of Northwestern University Library's Large Image Support		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	7	29.17%
Below average (2)	6	25.00%
Average (3)	6	25.00%
Good (4)	5	20.83%
Excellent (5)	0	0

Northwestern University Library's Feedback/Comments:

2	no maximize function. could not drag the image with left click, which is a nice function. simple an quick
5	slow, bad user interface, buttons should be easier and more clear
14	It was just luck I clicked the large image and it zoomed in. No instructions about what to do or what to expect. No controls - bit odd.
18	Did not like this one at all. Want the "feedback" and interaction to be on the same image, not two seperate ones.
19	Again, I am not sure what this "Rate the feedback mechanisms provided by the zooming function:" means.
24	At first it wasn't clear how to zoom out of the image.
25	Not very interactive.

1.4 Survey results for the Large Image Support of the Microsoft Seadragon (Non-Institutional Repository based solution):

Field Summary for 001:		
Were you able to zoom in on the image from Microsoft's Seadragon Website?		
Answer	**Count**	**Percentage**
No answer	0	0
Yes (Y)	24	100.00%
No (N)	0	0

Field Summary for 002:		
Did you need to install additional software to be able to zoom in on the image?		
Answer	**Count**	**Percentage**
No answer	0	0
Yes (Y)	1	4.17%
No (N)	23	95.83%

Field Summary for 003:		
Rate how successful you were in zooming-in on the image:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	0	0
Below average (2)	1	4.17%
Average (3)	1	4.17%
Good (4)	8	33.33%
Excellent (5)	14	58.33%

Field Summary for 004:		
Rate the control mechanisms that enables you to zoom-in on the image:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	0	0
Below average (2)	2	8.33%
Average (3)	1	4.17%
Good (4)	8	33.33%
Excellent (5)	13	54.17%

Field Summary for 005:		
Rate the feedback mechanisms provided by the zooming function:		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	2	8.33%
Below average (2)	0	0
Average (3)	4	16.67%
Good (4)	9	37.50%
Excellent (5)	9	37.50%
Field Summary for 006:		
Rate the response speed of the zooming functionality?		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	0	0
Below average (2)	2	8.33%
Average (3)	0	0
Good (4)	5	20.83%
Excellent (5)	17	70.83%

Field Summary for 007:		
Rate the effectiveness of the Microsoft's Seadragon Large Image Support		
Answer	**Count**	**Percentage**
No answer	0	0
Poor (1)	1	4.17%
Below average (2)	1	4.17%
Average (3)	2	8.33%
Good (4)	8	33.33%
Excellent (5)	12	50.00%

Microsoft Seadragon Feedback/Comments Summary:

2	easy to use. maximize function, yeah! visual feedback to see where you are in relation to original image would make this the best of the lot - near perfect for me.
5	Microsoft sucks!
8	just need a navigator to know where you are
14	Like the motion on dragging. Nice full screen option, could have been bigger in the frame before the full screen option. Best so far.
15	Only problem here was it took about 5 minutes to load.Otherwise, excellent image zoom function.
18	This one is great! Fast response as well,without page reload!
19	Didn't understand this question "Rate the feedback mechanisms provided by the zooming function:" So I answered as poor, which will probably skew your results.
21	They could have provided a google-maps style zoom-meter, but beyond that, the Ajax seadragon looks pretty good.
24	Intuitive input and very fast loading times.
25	Very lagged.

26	This system caused Firefox to hang for 2 minutes at one point (shortly after toggling full screen). I was just about to kill Firefox and start again. This is running on Ubuntu on a machine with fairly decent specifications.

Appendix B

1. First Prototype Assessment Results

	YES	NO
Image Processing		
1.1 - In the Open Window Double Click on DeepZoomer.exe	24	0
1.2 - In the "Browse For Folder" Dialogue Window - Select Computer, Then Local Disk (C:), Then Maps, Then Select the OK button. (The Browse for folder dialogue window will disappear - Wait a minute or two until a window appears)	22	2
1.3 - A Window will appear with the message: "Image Processing Complete" - Close this window.	24	0
2.DSpace Batch importing		
2.1 To open the Command prompt: - Click the Round Start Button Bottom Left Corner - Choose Run, then cmd, then OK Button	24	0
2.2 Type the Following in the black window: cd c:\dspace\bin (Press Enter)	24	0
2.3 After C:\dspace\bin insert the following command: (Copy the text line below and right-click in the Command Prompt to Paste) dsrun org.dspace.app.itemimport.ItemImport -a -e marius-nel@hotmail.com -c 123456789/326 -s "C:\maps" -m "C:\maps\mapfiles\mapfile.txt" Press Enter to run the command. Close the Command Prompt Window after the program completes. (When it returns to the C:\dspace\bin)	20	4
3.View the Large Image in the Browser		
3.1 Open Firefox (Click the Firefox tab on the taskbar)	23	1
3.3 In the right side bar under Recent Submissions - Choose Africa	24	0

3.4 Click on the Africa.html link in the grey "Files in This Item Box"	24	0
3.5 Click on the image displayed to Zoom in on the map. - You may also Click and drag to move the image left and right. - You may also use the mouse scroll wheel to zoom in and out of the image - There are also controls located in the window to enable you to inspect the image (Please feel free to use these)	24	0
Comments/Feedback		
1 - Change cursor to hand to indicate panning control availability. - Also, set maximum zoom-out level to fill the image-viewing window.		
2 - The sliding control does not respond to small incremental changes. - Group the tools in one area.		
5 - It's fun zooming in and out.		
12 - That was easy enough.		
Notes		
1.2. The waiting period confuses people. Majority of the people started clicking around before the prototype completed the image processing. - Two people failed to find the right directory		
2.3. Three people had difficulties with copying and pasting the batch import command. They tried the "control+v" keyboard short-cut which does not work in the command prompt window. - One person failed to copy the last character of the command and had to redo this step.		
3.5. One person tried to pan left and right whilst the image was zoomed out to maximum level. Panning only works when the image is zoomed in to a level that allows panning.		

2. Second Prototype Assessment Results

	YES	NO
1.Image Processing	24	0
1.1 - In the Open Window Double Click on DeepZoomer.exe		
1.2 - In the "Browse For Folder" Dialogue Window	23	1
- Select Computer, Then Local Disk (C:), Then Maps, Then Select the OK button		
(A Progress bar will appear to indicate that the program is busy processing images. After the image processing is complete a new form window will appear with two text fields)		
1.3 - Type (or copy and paste) the following in the first text field labeled "Collection Handle": 123456789/121	24	0
1.4 Type (or copy and paste) the following in the first text field labeled "Collection author email": mars@rocknroller.co.za		
1.5 Click Submit		
(A window with a black background will appear indicating the progress of the DSpace batch import process)		
1.6 When the batch import process has completed a window will appear with a message that reads Image Processing completed" - Close this window		
2.View the Large Image in the Browser		
2.1 Open Firefox (Click the Firefox tab on the taskbar)	23	1
2.2 Refresh the current window	24	0
2.3 In the right side bar under Recent Submissions - Choose Africa	24	0
2.4 Click on the Africa.html link in the grey "Files in This Item Box"	24	0
2.5 Click on the image displayed to Zoom in on the map.	24	0
- You may also Click and drag to move the image left and right.		
- You may also use the mouse scroll wheel to zoom in and out of the image		
- There are also controls located in the window to enable you to inspect the image (Please feel free to use these)		
Comments/Feedback		
1 - That was easy		

5 - loved the full-screen view		
5 - The image update is a bit slow		
12 - Smooth zooming effect		
17 - The blurring between the zooming levels was “slightly disorientating”.		

Appendix C

1. Prototype Performance test machine specifications

1.1. Operating System

Microsoft Windows XP
Media Center Edition
Version 2002
Service Pack 3

1.2. Computer

Intell® Pentium® D
CPU 280 GHz 2.79 GHz,
2GB RAM