UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

COMPUTER SCIENCE HONOURS
FINAL PAPER
2015

Title: Towards an Extensible Digital Cultural Heritage Archive

Author: Noxolo Mthimkulu (MTHNOX003)

Project Abbreviation: 3ARCH

Supervisor: Hussein Suleman

Category	Min	Max	Chosen
Requirement Analysis and Design	0	20	20
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	0
System Development and Implementation	0	15	15
Results, Findings and Conclusion	10	20	13
Aim Formulation and Background Work	10	15	12
Quality of Paper Writing and Presentation	10		10
Adherence to Project Proposal and Quality of Deliverables	10		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	0	10	
Total marks		**80**	**80**

Towards an Extensible Digital Cultural Heritage Archive *

Noxolo Mthimkulu † ‡
University of Cape Town
Rondebosch
Cape Town
mthnox003@myuct.ac.za

ABSTRACT

Digital cultural heritage archives provide a mechanism for cultural heritage preservation and wide-spread accessibility of the artefacts contained. The project resulted in the implementation of querying, exploration and personalisation archival services. The design of the services was focussed on concepts of rich-prospect browsing and the archival capabilities available with collections containing rich metadata. An extensible archive was achieved by performing automatic generation archive files without the need for system re-implementation. The extensibility was aided by the system's ability to allow for archive specific service configuration. Implementation of an extensible, configurable digital cultural heritage archive was achieved by adopting the processes of the Crystal Clear Agile software development methodology and the implementation concepts contained in Feature Driven Development.

1. INTRODUCTION

Cultural heritage preservation has migrated from physical archives to digital heritage archives. Digital archives manage and preserve multimedia information [20] while providing timeless accessibility to digitised artefacts [13].

The project was aimed at the introduction of a digital cultural heritage archive to house three distinct collections pertaining to the Western Cape.

1.1 Project Description

The Centre for Curating the Archive (CCA) of the University of Cape Town's Michaelis School of Fine Art [11] is responsible for collecting, curating and digitising various collections. The CCA makes these collections accessible to artists, scholars, students and other community members by providing Web access, publications and hosting events and exhibitions to showcase the materials conserved. Their collections include collections comprising artefacts centred around three distinct historical events occurring in the Western Cape: the Sequins, Self and Struggle archive, a collection containing multimedia objects from the Miss Gay Western Cape and Spring Queen beauty pageants; the Harfield Village collection, an aggregation of artefacts about the forced removals of the Claremont residents; and Movie Snaps, a collection of photographs taken in and around central Cape Town before and after apartheid.

1.2 Project Aim

The project's objective was to digitally preserve the cultural heritage presented within these archives and increase the accessibility of the information by providing online access. In addition, the project aimed to encourage users to contribute to the collections, thereby growing the collections.

Access to the multimedia items was provided through search and browse, history and personalisation, exhibition, maps, download and upload archival services.

Together with providing access to cultural heritage artefacts, the project aimed to investigate an extensible archival solution that allows for the dynamic allocation of available archival services to the existing collections. Furthermore, the configurable archive was also to allow for the introduction of new archives without system re-implementation.

This paper focusses on the background considerations, design process and implementation procedure carried out for the implementation of the search,browse and history and personalisation services. Furthermore, the paper discusses the design considerations and implementation strategies adopted gearing towards an extensible digital cultural heritage archive.

2. BACKGROUND

2.1 Digital Libraries

2.1.1 What is a digital library

Digital libraries are often seen as large databases, applications on the internet or automated versions of conventional libraries [8]. Digital libraries mirror conventional libraries

*http://personalhistories.cs.uct.ac.za/three_archives/

†Project website containing literature survey, project proposal and software engineering documentation: http://pubs.cs.uct.ac.za/honsproj/2015/

‡Github repository source code: https://github.com/noosrat/three_archives

by providing access to information, ensuring comprehensive organisation of information and preservation of work [8].

Digital archives manage and preserve multimedia information [20], providing timeless access to digitised artefacts. Digital cultural heritage archives are designed for use by cultural heritage institutions, cultural heritage professionals, academics and scholars, tourism users and the general public [7]. Digital archives have the potential of impacting teaching and learning [4]; where information is provided to users through archival services [2].

2.1.2 *Digital libraries and their services*

Digital libraries are required to have services allowing for the searching, browsing, selecting, grouping and often personalisation of the presentation of the digital objects contained [26]. Services available in digital archives indicate a convergence between a library and a museum but with archival content [13]. Furthermore, digital archives allowing users to contribute their own collections and services allowing users to create their own story are emerging [3].

The services provided are supported by digital repository tools that provide digital object storage and management services for digital libraries.

2.2 Repository tool: Fedora

Digital object repository tools provide an architecture for the storage and management of the digital objects. Various repository tools also provide archival services as an interface to the repository.

A literature survey on available digital repository tools resulted in the Flexible and Extensible Digital Object and Repository Architecture (Fedora) [27] being chosen as the project's repository tool. DSpace [18] and Omeka [23] were also considered, however, these are inflexible and would not allow full interface and service customisation as was required for the project.

The below examines the Fedora tool and its digital object storage and management.

2.2.1 *What is Fedora*

Fedora is an extensible digital content repository providing services for the storage, management and distribution of digital objects [19].

2.2.2 *Fedora Object Model*

Digital objects are contained in the repository layer of a digital archive and are used to manage the digital material [1]. The Fedora Object Model is at the core of the repository's architecture and is defined in Fedora Object XML (FOXML). FOXML is Fedora's XML representation of the digital objects, the object content and metadata.

The Fedora Digital Object

Figure 1 is a depiction of a Fedora Digital Object. The digital object comprises of a persistent identifier (PID) for unique identification of the object, system defined object properties used to manage and track the objects, and datastreams representing the object's content [30]. Every Fedora

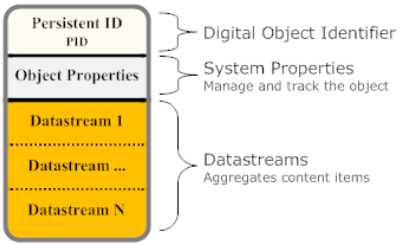

Figure 1: Components of a Fedora Digital Object[30]

digital object has a primary Dublin Core datastream conforming to the Dublin Core metadata schema [30]. Metadata is crucial in digital libraries as it allows for the organisation of the objects by good indexing, storage and access of objects and preservation [14].

Fedora allows for user-defined datastreams that contain the content. These items can be of any format and can be stored either within the repository or at any other specified location [30].

2.2.3 *Fedora Service Framework*

The Fedora Service Framework is where the Fedora digital objects are managed. The framework contains services that interact with one another and with the objects [19]. Fedora provides basic search and browse services which are exposed through a HTTP RESTful API [31]. The Fedora architecture allows for new, external services to be developed and integrated into the framework. These new services are able to leverage off of the existing Fedora services [19].

Fedora provides a Web interface for an administrator where ingestion and management of objects can be conducted [32]. A basic searching interface is also provided, allowing for tailored searches according to specific metadata fields [29]. The functionality exposed via these interfaces can be achieved programmatically through communication with the HTTP RESTful API provided.

The RESTful API provides access to the objects within the repository [34]. The retrieval methods exposed via the RESTful API include *describeRepository, findObjects, getObjectProfile* and *getDatastreams*. Methods to create and ingest digital objects include *addDatastream, addRelationship,export*,*getDatastream*, *getRelationships*, and *ingest* [19].

2.3 Digital Archive Services

Digital archives facilitate exploration of multimedia items contained and provide a personalised exploration experience. The following outlines the search, browse and personalisation services present in digital archives.

2.3.1 *Search*

Searching and information retrieval functionality is standard functionality in digital libraries and there are continuously new methods being developed for indexing and the retrieval of relevant information for the user [41]. Improvement in querying and browsing mechanisms is important and mechanisms involving searching outside of the digital object metadata need to be considered [6].

Digital archives implement multi-lingual and language independent retrieval where the querying involves querying a thesaurus [6]. An implementation of a thesaurus querying mechanism aids users in building queries, which assists in the users' struggle with locating relevant keywords [38].

2.3.2 Browse

Browsing functionality allows users to understand the collection and familiarise themselves with the contents and the structure of the repository before they can perform specific queries [41]. The design of advanced multimedia interfaces needs to be carefully considered in order to ensure that the interfaces and collections can be easily exploited in research, teaching and other environments [6].

The need for a well-organised display has led to the concept of a rich-prospect browser; it provides the user with a clear overview of what is available in the collection [36]. This assists the user in specific searches as it brings awareness and familiarity with the entire collection [36]. The rich-prospect browser maintains the following principles: a meaningful representation of each multimedia object presented on the primary page; the user should be able to reorganise the representations dependent on what they wish to browse and the organisation of the information should be relevant to the user; the metadata of the objects assists in the manipulation of the view; and the representations provided allow for more meaningful information to be retrieved with more than one representation available [36] [35].

A combination of rich-prospect browser concepts and focus as to what is representable using the metadata fields and values was considered during the design and implementation process of the search and browse services.

2.3.3 History and Personalisation services

Recommendation services consider recent queries against collections in order to provide the user with a personalised experience. The services focus on the system's ability to learn material related to what is important and relevant and to monitor user actions in order to determine this [6].

Recommendation service functionality is necessary in order to minimise the users' effort and prevent repetition when interacting with the system and requesting familiar resources. Information obtained from the user as a result of their activity aids in the personalisation of the experience [2]. Digital archives implement personalisation systems by categorising the users' interest and by applying recommendation algorithms [2].

The awareness and personalisation services implemented in the project focussed primarily on updates since the users' last visit and trending keyword searches and browses.

Archival service implementation involved adoption of a Software Development methodology in order to aid the management of the project as well as of the code base. The paper continues with a discussion of the Software Development process adopted for the implementation of the search, browse and history services implemented.

3. SOFTWARE DEVELOPMENT LIFE CYCLE

The software development methodology approach adopted involved the combination of two agile methodologies: the Crystal Clear methodology and Feature Driven Development (FDD). An agile software engineering approach was chosen in order to ensure that features are developed incrementally and that the team always had a working product aided by project management and tracking.

The Agile methodologies chosen both involved an incremental, iterative approach with commonalities in some of the processes enforced.

3.1 Crystal Clear methodology

The Crystal Clear methodology comes from the Crystal family of agile development methodologies [9] and has been seen as lightweight and adaptable with early and frequent delivery of working software with user involvement [16].

Crystal Clear enforces a process framework with a defined process to be followed with details defined by the team [16]. Crystal Clear's flexibility provided a good skeleton to follow for the project implementation with documentation adjustments and the inclusion of the Feature Driven Development practices.

The Crystal Clear process involves forming a team and conducting a feasibility analysis to arrive at an initial plan [16]. The second element involves delivery cycles where the team updates and refines release plans while implementing subsets of requirements. These cycles include frequent integration of the software and constant delivery of the product to users [16]. The methodology is continuously reviewed after the iterations [16]. The final step of the Crystal methodology involves acceptance testing and deploying to the correct environment.

The team incorporated weekly team meetings with the supervisor into the Crystal Clear methodology. The meetings provided a mechanism for progress feedback and guidance on how to conduct user testing and elicit further requirements. Object oriented requirement documentation and meeting minutes were included. The team also introduced a communicator who was responsible for communicating with the client and users.

The Crystal Clear methodology was used for the management of the project; and the development of the project leveraged off of Feature Driven Development.

3.2 Feature Driven Development

The actual implementation of the project leveraged off of principles of Feature Driven Development [37]. The project life cycle was iterative with all considerations of Crystal Clear methodology including the five steps of Feature Driven Development. Figure 2 depicts the FDD process which involves developing an overall model, building a features list, planning by feature and designing and building by feature.

These steps were carried out, in combination with the Crystal Clear methodology, for each iteration of the develop-

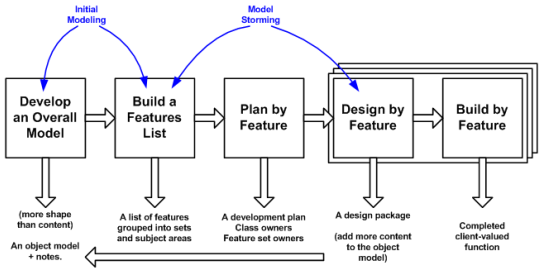

Figure 2: The FDD Project Lifecycle [37]

ment process. Constant user feedback and evaluation was obtained in order to guide the built features and to refine requirements further. The FDD steps involve developing an overall model and exploring the structure of the problem followed by the construction of a features list [25]. This list acts as a backlog of the system and each team member is assigned a feature and particular classes to work on. The *plan by feature* element speaks to which classes would be dedicated per feature and the *design and build by feature* are the implementation phases of the software. Following the design is iterative planning that takes place as a result of the new understanding obtained. The process includes unit testing, code review and user feedback and occurs iteratively until project completion [25]. This iterative process mirroring the Crystal Clear methodology.

4. REQUIREMENTS ANALYSIS AND SYSTEM DESIGN

4.1 Requirements Gathering

This section describes the requirements gathering process carried out throughout the system implementation. The results obtained from each of the sessions are expressed in Section 5.5. The requirements gathering for the project was done in an iterative manner to mirror the software development methodologies chosen. The first iteration was preceded by an extensive requirements gathering phase during the inception of the project where requirements and understanding of the project was obtained from the CCA. The features were outlined and an understanding of technologies to be used throughout the implementation was gained by conducting a literature survey.

The initial meeting with the client, the CCA, resulted in a high-level overview of the functionality required of the system. The CCA indicated that the digitised artefacts were being stored centrally on hard drives located internally and that the public could not access the archives. In addition, the inaccessibility of the archives resulted in the public having limited knowledge about the existence of the collections.

The functionality required by the CCA was discussed with the project supervisor and a finalised features list was constructed. The features included search and browse, history and statistics, exhibitions, commenting and annotations, map interface functionality and downloads and uploads. The features list coincided with the feature list expected in FDD. The Crystal Clear and FDD methodologies both expect iterative implementation of software. The team

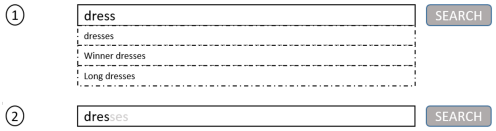

Figure 3: Lo-fi prototype for search auto-completion interface

and project supervisor assigned features per iteration with the project spanning three iterations.

4.1.1 Iteration 1

A contextual enquiry was conducted in order to elicit understanding as to what the typical system users were familiar with and how they normally interacted with technologies to be incorporated into the archives.

4.1.2 Iteration 2 and Iteration 3

Subsequent requirements gathering occurred in the form of a focus group conducted with both the clients and with students familiar with digital libraries. The requirements gathering conducted with the client allowed for the refinement of the expected archival functionality and the focus group with the students assisted in the interface design of the application.

The focus groups involved presenting questions for discussion that would gauge comfort and understanding of the functionality. These questions and results this form part of the software development documentation.

The design focus group involved lo-fi prototype interface presentations where feedback was obtained from the users as to their interface preferences would be when experiencing searching and browsing functionality. Figure 3 is a sample representation of the lo-fi prototypes proposed during the design focus group. In this diagram we see two different representations of how auto-completion could be presented to the user: (1) indicates auto-completion in the form of a drop down, and (2) indicates auto-completion of the word within the search box. The participants agreed that they preferred the first method of auto-completion with a list of the matched items but would also like to have the word completed while they typed as in (2).

4.2 System Design

The distinct archival services and features required resulted in the choice of a component-based three-layered architecture. The component-based architecture allowed for separation of and distinction between features within the codebase. A three-layered architecture considering the structure of the application. Figure 4 depicts a high-level view of the design of the system and the architecture followed. The interface layer is where the client would communicate with the system. The interface development involved developing an interface for the Harfield Village archive as well as interfaces for the search,browse and history services. The service layer is where the business logic pertaining to archival services was implemented. The back-end layer is where the storage and management of digital objects occurred.

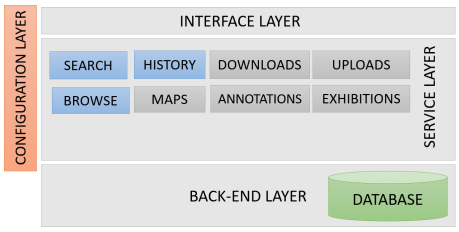

Figure 4: High level system architecture

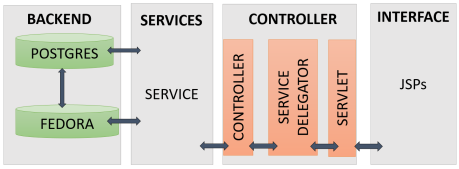

Figure 5: System class model overview

The configuration layer included in Figure 4 aided in the system's extensibility. The configuration layer governs the front-end and the services of the archives. This layer ensures that each distinct archive reflects the services which have been made available to it. The configuration distinction within the back-end layer lies with prefix of the objects' persistent identifiers.

Figure 5 represents the class level identification of the three-layered architecture. The figure places emphasis on the componentised design with each feature following the same layout. Each feature would have a *JSP* as the interface and each *JSP* would communicate with the system through a single *Servlet*. The *Servlet* is responsible for dispatching the response to the *JSP*s and for passing control of the request to the *ServiceDelegator*. The *ServiceDelegator* delegates control to the relevant service's *Controller* dependent on what was contained within the request URL. The service *Controller* then calls the relevant *Service*. The services communicate directly to the digital object repository and database. Figure 5 represents a high-level view of the the project's system class model diagram.

5. SYSTEM DEVELOPMENT AND IMPLEMENTATION

5.1 Workstations

The below is a discussion of the tools and technologies used throughout the implementation of the system.

5.2 Workspace

Eclipse: Eclipse Mars [10] was used as the Integrated Development Environment for the project as it allowed for integration of both the build tool used and the code management tool used.
Apache Maven: Apache Maven 3.3.3 [5] was used to regularly build and package the project into a Dynamic Web Project artefact to be deployed into the server.
Apache Tomcat: Apache Tomcat 8.0.24 [40] was used as the application server and servlet container facilitating management of the servlets as well as of the application. The server also hosted third party software such as Solr and Fedora and facilitated the communication between the applications.
Git and Github: Git and Github [15] were used as a code management and repository tool. The tool allowed for focussed feature development by taking advantage of the branching capabilities provided.

5.3 Backend

Java: The Java™ SE Development Kit 8, Update 45 [24] was used for the development of the software product allowing Object Oriented Programming principles to be implemented. In addition, the Java Enterprise Servlet architecture was used in order to construct a Web application and to communicate with the server.
Fedora: Fedora 3.8.1 [33] was used as the repository to house and manage the digital objects.
Solr: Apache Solr 4.6.1 [39] was used to index the items in the archive and was used as a search engine for the items. The schema.xml was reconfigured to index only the Dublin Core fields. Using the Solr search engine resulted in objects which were better indexed ensuring more relevant results when querying the archives.
FedoraGSearch: FedoraGSearch 2.7 [28] assisted with the integration of Solr and Fedora. The Solr platform allowed for more indexing and retrieval of digital objects in a manner that Fedora alone does not provide. Fedora Gsearch provided a mechanism for these two technologies to communicate such that when an object is uploaded in the Fedora repository, FedoraGsearch is used to re-index items within the repository based on Solr indexing requirements.
PostgreSQL: PosgresQL[12] 9.3 was used as the database in order to store properties of the digital objects.

5.4 Frontend

Bootstrap: Bootstrap[22] along with JQuery[17] and HTML 5 were used in order to design and implement the user-facing interfaces of the application.
Browser Cookies: The application leveraged off of the browser cookies in order to successfully implement the history and personalisation services.

5.5 Implementation

5.5.1 System Overview

The system was designed with a three layered architecture with services and interfaces governed by the configuration layer. The three layered architecture has been depicted in Figure 5. The iterations below discuss the implementation of the search and browse, and history and personalisation services followed by a discussion of the configuration layer.

5.6 Iteration 1: Basic Search

5.6.1 Requirements Gathering and Features List

This iteration involved the requirements gathering process during the inception of the project and the contextual enquiry conducted as discussed in Section 4.1 and Section 4.1.1. The contextual enquiry was conducted with three people who were familiar with the collections. One participant had been involved with obtaining and digitising the items, another had been involved in curating the material and the final participant had used the archives for teaching purposes. The contextual enquiry did not elicit any new requirements

but assisted in understanding the participants' familiarity with the services to be implemented. The participants were all familiar with the use of search engines and browsing functionality and indicated that the main difficulty experienced was largely in navigation and awareness of their position within the archive throughout their searching and browsing. The participants also indicated that their existing method of accessing the archival content did not provide sufficient information for teaching purposes.

5.6.2 Planning by feature

The feature implemented during this iteration was the basic search service. Understanding the Fedora repository tool was a large part of what was required for this feature. It was necessary to install and configure the Fedora repository tool for the environment in order to allow for storage of the digital objects. In addition, understanding the communication with the repository tool via the RESTful API was necessary. This communication needed to be grasped by first understanding the services provided by the repository and accessing objects via the command line. It then became evident that a Java client would need to be built in order to allow the application to communicate with the repository.

5.6.3 Design and Build by feature

The design process involved understanding the components necessary to build a complete Fedora client to speak to the API provided. This process involved extensive design of what objects would be needed to communicate with the repository as well as actual implementation. The attributes for the domain objects for the *FedoraDigitalObject* and the *Datastream* mirrored those of the actual repository digital objects.

The process of a successful search involved the user accessing the service through the search interface. This can be observed in Figure 6. The image is an extract of the full sequence diagram and illustrates the process that occurs when populating the object profile of a *FedoraDigitalObject*. The precondition is that the search terms have been entered and the interface would then propagate the request to the *SearchController* in order to communicate with the digital repository. The keywords entered using the interface would communicate with the *FedoraCommunicator*, which would build the necessary query in the form of a *FedoraGetRequest*. The keywords and query parameters would be populated within the *FedoraGetRequest*. After the request has been built, the *FedoraClient.execute(FedoraGetRequest fedoraGetRequest)* method is then called. This method propagates the request to the RESTful API which returns a response. The *FedoraXMLResponseParser* is then responsible for parsing the XML response obtained from the repository into the front-end *FedoraDigitalObject* to be made available on the front-end. This is illustrated by Figure 6. Figure 6 illustrates the beginning phase of populating the digital object. Following this, a *FedoraGetRequest* would be constructed to query the repository for the object's datastreams. The process would continue in the same way as described above and in Figure 6. This process would be carried out for any request being made to the digital object repository.

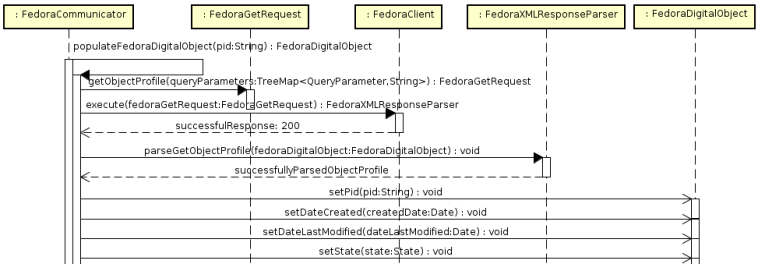

Figure 6: Sequence diagram of populating a Fedora Digital Object object profile

5.6.4 Evaluation

The first iteration was implemented as a feasibility demonstration to allow for exploration and understanding of the tools to be used throughout the service implementation. The feasibility demonstration was evaluated by presenting the functionality to the project supervisor and the second reader. The successful retrieval of digital objects indicated full understanding of the Fedora tool and communication with the API. Automated tests were written for the *FedoraGetRequest* to ensure that the requests to be sent to the HTTP RESTful API were correctly structured.

Reflection of the software process, as per the Crystal Clear methodology, indicated that the writing of unit tests upfront would be infeasible given lack of experience and time constraints. The decision was made to write automated tests post implementation of each feature. Re-evaluation of the software process also resulted in the conclusion that further requirement refinement in the form a a formal focus group was necessary. In addition, weekly progress reports would be required in order to track the status of the project.

5.7 Iteration 2: Advanced search and browse

5.7.1 Requirements Gathering and Features List

Requirements for the advanced search and browse were obtained as discussed in Section 4.1.2. A list of questions was drafted for the focus groups in order to refine understanding of the required features. A set of lo-fi prototype sketches for possible interfaces was presented to participants in one of the focus groups.

The design focus group was conducted with third year computer science students. The participants were presented with a variation of lo-fi prototypes for possible searching and browsing interfaces and were encouraged to provide feedback and comments about their preferences. The prototypes and detailed feedback obtained form part of the software engineering documentation. The focus group with the client identified that the client found difficult with navigation of collections in digital archives and digital libraries. The client indicated that the items within the collection should be categorised according to date, title and format and that specific searching by media-type, event, academic paper, location and exhibition would be desirable.

Both focus groups resulted in evidence that users prefer auto-complete and being guided throughout their search as often they feel as though time is being wasted on fully typ-

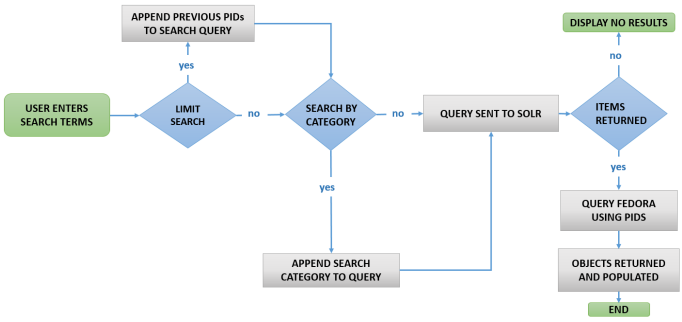

Figure 7: Flow chart of advanced system search

ing out the search terms with disappointment occurring after effort has been expelled to enter the search terms but no results are returned. The focus groups also brought awareness to users' confusion with the scope of their search once they have already been browsing and searching a digital archive. The client also indicated the lack of notification as to where they are in their searching or archives resulted in a dissatisfying experience.

5.7.2 Planning by feature

The second iteration saw the introduction of browse as well as advanced searching functionality. The planning involved obtaining information about how to successfully incorporate the Solr indexing and searching tool with the Fedora tool in order to achieve a better search on a well indexed repository. The introduction of the Solr tool resulted in further refinement of the Fedora Client built as well integrating the Fedora client and repository with the Solr tool in order to take advantage of Solr's indexing. During the planning phase it became evident that development of a Solr client would not be necessary as queries could be conducted through the existing Solr API provided. The results of these queries in the form of an XML or json file. The FedoraGsearch tool was necessary in order to provide successful integration and configuration of the Solr and Fedora tools within the tomcat server.

5.7.3 Design and Build by feature

Analysis of the results obtained from search Solr resulted in a design where the system would query the Solr search engine for the identifiers of the digital objects matching the search terms and then use these results to obtain the items from the Fedora repository. The introduction of the Solr search engine required a *SolrCommunicator* to facilitate communication with Solr using the SolrQuery API provided by Apache.

Figure 7 illustrates the process flow for the advanced search functionality. The advanced searching involved the introduction of capability to limit the search results to a specific scope as well as to search within a specific category. The advanced searching also saw the introduction of auto-completion. The auto-complete words were sourced from the metadata used to build up the archive. This ensured that the auto-complete always matched items that existed within the archive.
The browsing service placed focus on identifying the most relevant categories to group objects into in order to allow for a rich-prospect browser experience. The implementation involved including all the Dublin Core fields as categories that the user could browse by. The design decision involved only presenting the user with the categories which have elements within them for browsing purposes, hence, empty categories did not appear on the interface.

5.7.4 Evaluation

This evaluation process involved obtaining feedback from experts in the Digital Libraries field. The participants were presented with the search and browse functionality and were asked to freely browse the archives and give any feedback from that experience. This was followed by the participant being guided around the interface by the facilitator and prompted to perform certain tasks. The participants were encouraged to talk aloud throughout the process in order to obtain on-the-fly feedback of their experience.
The feedback obtained from these users was primarily interface based. The feedback outlined the need for a sorting mechanism for the results and that the interface should represent the browsing categories as a list and the subcategories as thumbnails rather than two lists as the side navigation bar became cluttered. The participants were unsure of the search scope limiting functionality and indicated that it needed to be more apparent in the label of the check box.

Automated tests for the *SolrCommunicator*, *Search* and *Browse* classes were implemented.

Evaluation of the development process indicated that the time constraints meant that more work would need to be put in order to successfully complete the project and that testing and client interaction needed to be organised in advance in order to prevent delays.

5.8 Iteration 3: History and Personalisation and Configuration layer

5.8.1 Requirements Gathering and Features List

The history and personalisation services and the configuration layer were implemented during the final iteration. In addition, interface finalisations were to be made. The requirements for the final iteration had been gathered during the aforementioned focus groups and contextual enquiry. The feedback obtained from the focus groups pertaining to the history and statistics functionality was that users would find this functionality useful, however, there was concern about privacy and participants indicated that their browsing should not being reconstructible.

5.8.2 Planning by feature

Development of the history functionality was focussed on understanding cookies and the implications and options available with the use of cookies in a website. It was decided that the configuration layer would be a tool that would be accessible by somebody familiar with HTML and would have access to the actual code base but would not need to re-implement the system in order to introduce a new archive. The configuration layer required understanding of when services became available in order to toggle them at that point for each respective archive.

Figure 8: Harfield Village tag cloud

```
archive.name=Harfield Village
archive.landingpage.image=/images/harfield.jpg
archive.multimedia.prefix=hv
service.searchandbrowse=true
service.historyandstatistics=true
service.exhibitions=true
service.uploads=true
service.maps=true
service.downloads=true
service.annotations=true
```

Figure 9: Harfield Village properties file

5.8.3 Design and Build by feature

Personalisation design involved using cookies to obtain information as to when the user had last visited the archive. Recording the users' last visited date would allow for a comparison of the digital objects' "dateLastModified" property with the specific cookie date in order to retrieve items that have been modified since the user's last interaction. These updates were categorised as per the browsing categories and the interface allowed the user to filter out the updates dependent on the category selected.

Functionality to indicate trending searches was implemented by the use of a tag cloud. The system would keep record of the searches and browsing conducted throughout the archive by all the users of the archive. This information was recorded in a text file stored on the server and the content of the text file was used to construct the tag cloud. A sample tagcloud from the Harfield Village archive can be observed in Figure 8 where collection is observable as the most searched term.

The configuration layer saw a two fold implementation: one where the creation of the new archive was automated and the second implementation which involved the management of the archival services dependent on which archive was being accessed. The implementation involving which services to enable or disable was governed by the use of a properties file with the structure as in Figure 9. Each archive would have one of these files where their properties and services would be specified. The application would then enable or disable the services dependent on what was made available in the properties file. The implementation of the introduction of a new archive to the system adopted a process of automatic generation of new files and regeneration of existing files which would need to represent the information of the new archive. The archive configuration process involved: the generation of a properties file where the user running the configuration suite would be prompted for the values; the generation of the landing page for the application to include a navigation option to the new archive; and a blank home page with the generic navigation bar indicating the services available would be generated for the new archive.

5.8.4 Evaluation

The evaluation of the configuration layer was done by unit tests and verification of whether the correct files had been automatically generated. The unit tests also involved checking whether the correct interface elements indicating the enabled and disabled services were present.

The evaluation of the history feature and the completed search and browse functionality took place with an evaluation of the system in the form of user tests. The final evaluation process as well as the results, findings and conclusions are addressed below.

6. EVALUATION

The final evaluation took place at the end of the third iteration in three different stages of testing with different user groups. The testing and evaluation of the system involved user acceptance testing, testing with experts in the field as well as usability testing. These will be further discussed below and the findings obtained and conclusions drawn will follow.

6.1 User Acceptance Testing

The User Acceptance Testing (UAT) for the application was conducted with three participants from the CCA. The manner of the user acceptance testing involved the users being presented with the application and instructions read out to the users while the facilitators observed and encouraged the users to communicate aloud throughout the session. Each participant was required to test specific requirements and system features pertaining to the initial search and browse and history requirements while the facilitators observed. Each individual offered feedback in terms of requirements based testing involving feature acceptance and feedback in regarding system usability.

Table 1 indicates the overall requirements with sub-features. These requirements were assessed by the client. The client was asked to test certain aspects of the application in order to ensure full exposure to all the requirements and to gauge acceptance of the requirements and whether they have been fulfilled. As per Table 1,the CCA accepted all of the functionality tested.

Observations and Feedback

6.1.1 Search and Browse

The client understood the text-based search and assistance from the auto-complete functionality. The client found the auto-completion useful because the values stemmed from the metadata in the archives versus from a general dictionary. The values stemming from the archive metadata immediately gave the client a sense of what is contained within the archive. The "Using links/search tags to navigate" entry in Table 1 pertains to the search tags which are returned when the search results are returned. These tags are returned on the basis of similarity to the search entered. The client understood this functionality and intuitively clicked on the tags to continue navigating through similar searches.

The sub-requirements under browse were all intuitive to the participant. The client also indicated that being able to

Table 1: Requirements based User Acceptance Testing results

Functionality	Pass or fail
Search	
Text-based search	Pass
Auto-complete	Pass
Limiting searching scope	Pass
Using links/search tags to navigate	Pass
Sorting results	Pass
Searching using categories	Pass
Browse	
Browsing the images	Pass
Browsing using the categories	Pass
Viewing metadata for each item	Pass
Use of metadata to assist browsing	Pass
History	
Tagcloud	Pass
Filtering recent results	Pass

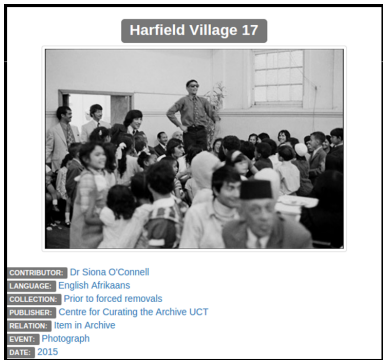

Figure 10: Dialog after selecting archive item

browse by the Dublin Core fields was useful as it provided a thorough categorisation of the archival items.

Figure 10 is what is presented on clicking a thumbnail on the browse interface. The functionality in Table 1 reading "use of metadata to assist browsing" was tested by prompting the user to click on a thumbnail and asking them to comment on and explore the metadata elements presented in the resulting pane. It was not immediately obvious to the user that they could click on the links representing the metadata values to navigate them to searching that specific item. Once this was understood the client agreed that this could be learned and would be understood after spending time on the dialog and eventually hovering over the actual metadata value. The client approved this functionality as it provided an aspect of easily accessing items that are related.

6.1.2 History and personalisation

The client was satisfied with the history and personalisation functionality. The comments made about the "recently updated items" aspect involved that it was a huge benefit since people often have a perception that archives are complete whereas they are often still being curated and still growing. The updates view allows users to grasp the concept of a developing archive. The client expressed that the tagcloud contained within this functionality was useful as a trend indicator for all users of the archive and that the use of it is intuitive.

The client navigated the archive with ease and was pleased with the layout and simplicity of the interface. The client also offered feedback in terms of a usability survey. The feedback obtained from the usability survey will be discussed in conjunction with the values retrieved from the usability testing Section 7.1.

Table 2: Usability tests

Criteria	Client	Expert	Student	Other
Usefulness	3.92	3.54	3.9	3.96
Ease of Use	4.09	4.15	3.62	3.73
Ease of Learning	4.25	4.58	3.92	4.17
Satisfaction	3.94	4	3.36	3.67
	4.05	4.07	3.7	3.88

6.2 Usability Testing

The Perceived Usefulness and Perceived Ease of Use of the system was evaluated by giving users a Measuring Usability with the USE Questionnaire [21] to complete after their interaction with the system. The users asked to rate the experience of the search and browse functionality by completing the questionnaire.

Four different groups of people were asked to evaluate the usability of the system: a group of three Digital Libraries experts, three employees of the CCA, a group of six Computer Science students and three randomly selected ordinary users. The groups were given the same questionnaire for feedback, however, the evaluation processes differed. The client and expert group were assessed in a manner such that instructions were verbalised and they were encouraged to think aloud throughout their experience with the system and provide comments and feedback.
The Computer Science students and ordinary users were given a list of tasks to complete using the interface and were required to fill in the evaluation questionnaire after their interaction. The distinction in testing mechanism was made in order to obtain comment-based qualitative feedback from the clients and experts given their experience with similar applications.

Table 2 presents and aggregation of the averages obtained from the questionnaire given to the participants on evaluation of the system. Each column represents one of the groups that were asked to evaluate the system. The questions asked as well as detailed individual results can be obtained on the website. The questions were categorised according to the four categories within the table. These questions allowed for the measure and understanding of the system's perceived usefulness and perceived ease of use.

The rating given ranged from 1 to 5 where 5 indicated strong agreement and 1 indicated strong disagreement.

Table 3: Usability tests

Criteria	Overall average
Usefulness	3.83
Ease of Use	3.89
Ease of Learning	4.23
Satisfaction	3.74

7. FINDINGS AND CONCLUSIONS

7.1 Findings Discussion

Findings can be observed in Table 2 and Table 3. The below discusses the results contained within these tables.

7.1.1 Usefulness

The usefulness rated an overall of 3.83 of 5 indicating that the participants, although often neutral about the usefulness were more in agreement with the usefulness of the functionality than not. The experts scored usefulness lowest whereas the other groups rated it above 3.9. Motivation behind why the client scored the usefulness highly would be that the search and browsing functionality of the system had been developed for them and to fulfil their specific needs therefore the questions under the Usefulness criteria were directly related to them. The experts expressed that the questions contained under the Usefulness category were unrelated to them. The experts responded with a neutral rating whenever the questions seemed inapplicable.

7.1.2 Ease of use

The overall rating for ease of use was 3.89 indicating close agreement with the functionality's ease of use. The client and experts both had an average rating greater than 4 for this criteria. This communicates that the two groups found the searching and browsing functionality easy to use. This may have been as a result of the client's experience with the project as a whole and the numerous evaluation, questions and prototyping that had been conducted with them. The expert participants had experience with similar applications and hence the functionality was intuitive to them.

The students and ordinary users rated it slightly below agreement. Students and ordinary users, although the students were computer students, may not be familiar with digital archives and navigating the collections within these.

7.1.3 Ease of learning

The experts rated the ease of learning highest gearing towards the strongly agree mark of 5. The client and ordinary users agree that the software was easy to learn. The students were close in agreement with the ease of learning of the system. A total average of 4.23 indicating that the the participants are in agreement that the functionality is easy to learn.

7.1.4 Satisfaction

The clients and experts were satisfied with the search and browsing functionality. The students and were more neutral about their level of satisfaction and the ordinary users agreed with the level of satisfaction. The level of satisfaction with the functionality could have been further influenced by the relevance of the system to the participants.

7.2 Conclusions

The project aimed to create an extensible digital cultural heritage archive to increase the accessibility of three distinct collections curated by the CCA. The project resulted in the successful implementation of searching and browsing and history and personalisation services with client acceptance and usability verified.

7.2.1 Process

The Crystal Clear and Feature Driven Development software methodologies resulted in the successful implementation of the features required within the system given the team size and time line required. The Crystal Clear also allowed flexibility in terms documentation produced and process followed.

7.2.2 Configuration Layer

The project aimed at developing a configurable, extensible archive. This functionality was successfully implemented with the being evaluated by the system's ability to introduce another archive to the system without complete reimplementation. This objective was successfully achieved with the extensibility being driven by auto-generation of files and the configuration governed by the configuration layer.

7.2.3 Features

The archival services were evaluated during various testing sessions and it became evident that expected functionality had been successfully implemented with the search and browsing functionality taking advantage of concepts of the rich-prospect browser to provide a fluid and easily understandable experience. The history and personalisation functionality was well understood and received.

7.2.4 Evaluation

Although the archival services were accepted and generally perceived useful and easy to use, the questionnaire chosen as well as the approach to the evaluation sessions between groups may have affected the results and rendered the results incomparable. This is since the experts and clients were verbally instructed and interacted with throughout the evaluation session whereas the students and ordinary users were given instructions to complete without interference from the facilitators.

A different survey would have also been more desirable to gauge the general usability of the system without relevance as to whether the participant had any prior knowledge or experience with the archives. Testing with community members and students who have been taught using the archive would have yielded different results and steered the entire development process in a different direction. This would have been a better reflection of the usefulness of the functionality and would have resulted in a more an archive better tailored to those users.

The objectives were achieved by following a Software Engineering methodology suited for the implementation of a configurable digital cultural heritage archive allowing for the storage, management and access of information representing the cultural heritage of minority groups of the Western Cape.

8. FUTURE WORK

8.1 Search and Browse

Future work for the services implemented would involve a multi-lingual archive for the search and content based image retrieval in order to prevent the archive relying on the metadata in order to fulfil searches made by the user. In addition, searching and browsing by exhibition could be a category to be included given that one of the archival services is the creation and viewing of exhibitions. Although the browsing did provide categorisation, the searching was not fully faceted as it did not provide the ability for multiple filters to be applied simultaneously.

8.2 History

Future work pertaining to the history and personalisation services could involve actual recommendation algorithms used to provide users with a personalised experience by observing the crowd and similar user profiles.

8.3 Configuration Layer

Future work for the configuration layer could involve being able to add additional services to the digital archive. Ensuring extensibility in terms of the addition of new archives but also in terms of new services.

9. REFERENCES

[1] W. Y. Arms, C. Blanchi, and E. A. Overly. An architecture for information in digital libraries. *D-Lib Magazine*, 3(2), 1997.

[2] H. Avancini and U. Straccia. Personalization, collaboration, and recommendation in the digital library environment cyclades. In *Proceedings of the IADIS International Conference Applied Computing (AC-04)*, pages 589–596. Citeseer, 2004.

[3] M. Barak, O. Herscoviz, Z. Kaberman, and Y. J. Dori. Mosaica: A web-2.0 based system for the preservation and presentation of cultural heritage. *Computers & Education*, 53(3):841–852, 2009.

[4] C. M. Bolick. Digital archives: Democratizing the doing of history. *International Journal of Social Education*, 21(1):122–134, 2006.

[5] J. C. M. M. Brett Porter, Jason van Zyl. Maven: Download apache maven, 2015.

[6] C.-c. Chen, H. D. Wactlar, J. Z. Wang, and K. Kiernan. Digital imagery for significant cultural and historical materials. *International Journal on Digital Libraries*, 5(4):275–286, 2005.

[7] G. Chowdhury. From digital libraries to digital preservation research: the importance of users and context. *Journal of documentation*, 66(2):207–223, 2010.

[8] G. Cleveland and I. U. Dataflow. *Digital libraries: definitions, issues and challenges*. IFLA, Universal dataflow and telecommunications core programme, 1998.

[9] A. Cockburn. *Crystal clear: a human-powered methodology for small teams*. Pearson Education, 2004.

[10] I. Eclipse Foundation. Eclipse downloads, 2015.

[11] C. for Curating the Archive 2015. Centre for curating the archive, 2015.

[12] T. P. G. D. Group. Postgresql: The world's most advanced open source database, 2015.

[13] S. Gwangjing-gu. A study of larchiveum data model for the design of digital heritage museum1. 2014.

[14] D. Hoe-Lian Goh, A. Chua, D. Anqi Khoo, E. Boon-Hui Khoo, E. Bok-Tong Mak, and M. Wen-Min Ng. A checklist for evaluating open source digital library software. *Online Information Review*, 30(4):360–379, 2006.

[15] https://github.com. Github, 2015.

[16] V. Inc. Agile methodologies for software development | versionone, 2015.

[17] jQuery Foundation jquery.org. jquery, 2015.

[18] M. Kurtz. Dublin core, dspace, and a brief analysis of three university repositories. *Information Technology and Libraries*, 29(1):40–46, 2013.

[19] C. Lagoze, S. Payette, E. Shin, and C. Wilper. Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6(2):124–138, 2006.

[20] B. Ludäscher, R. Marciano, and R. Moore. Preservation of digital data with self-validating, self-instantiating knowledge-based archives. *ACM Sigmod Record*, 30(3):54–63, 2001.

[21] A. M. Lund. Measuring usability with the use questionnaire. *Usability interface*, 8(2):3–6, 2001.

[22] J. T. Mark Otto and B. contributors. Bootstrap: The world's most popular mobile-first and responsive front-end framework., 2015.

[23] A. Morton. Digital tools: Zotero and omeka. *Journal of American History*, 98(3):952–953, 2011.

[24] h. Oracle Technology Network. Java se development kit 8 - downloads, 2015.

[25] S. R. Palmer. Feature-driven development (fdd), 2014.

[26] D. Paneva, L. Pavlova-Draganova, and L. Draganov. Digital libraries for presentation and preservation of east-christian heritage. In *Proceedings of the Second HUBUSKA Open Workshop âĂIJGeneric Issues of Knowledge TechnologiesâĂİ, Budapest, Hungary*, pages 75–83, 2005.

[27] S. Payette and C. Lagoze. Flexible and extensible digital object and repository architecture (fedora). *arXiv preprint arXiv:1312.1258*, 2013.

[28] G. S. Pedersen. Generic search service 2.7 - fedora framework services - duraspace wiki, 2014.

[29] S. Prater. Basic search - fedora 3.8 documentation - duraspace wiki, 2014.

[30] S. Prater. Fedora digital object model - fedora 3.8 documentation - duraspace wiki, 2014.

[31] S. Prater. Http apis - fedora 3.8 documentation - duraspace wiki, 2014.

[32] S. Prater. Fedora administrator - fedora 3.8 documentation - duraspace wiki, 2015.

[33] S. Prater. Installation and configuration - fedora 3.8 documentation - duraspace wiki, 2015.

[34] S. Prater and A. Benjamin. Rest api - fedora 3.8 documentation - duraspace wiki, 2015.

[35] S. Ruecker. Experimental interfaces involving visual grouping during browsing. *Partnership: the Canadian*

Journal of Library and Information Practice and Research, 1(1), 2006.

[36] S. Ruecker, M. Radzikowska, and S. Sinclair. *Visual interface design for digital cultural heritage: A guide to rich-prospect browsing*. Ashgate Publishing, Ltd., 2011.

[37] A. i. Scott W Ambler. Feature driven development (fdd) and agile modeling, 2014.

[38] A. Stafford, A. Shiri, S. Ruecker, M. Bouchard, P. Mehta, K. Anvik, and X. Rossello. Searchling: user-centered evaluation of a visual thesaurus-enhanced interface for bilingual digital libraries. In *Research and Advanced Technology for Digital Libraries*, pages 117–121. Springer, 2008.

[39] h. The Apache Software Foundation. Apache solr - apache download mirrors, 2015.

[40] h. The Apache Software Foundation. Apache tomcat - apache tomcat 8 downloads, 2015.

[41] C.-F. Tsai. A review of image retrieval methods for digital cultural heritage resources. *Online Information Review*, 31(2):185–198, 2007.