UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2020

Title: South African National Heritage Portal Prototype

Author: Alex Priscu

Project Abbreviation: HERIPORT

Supervisor(s): Hussein Suleman

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

HERIPORT: A Low-Cost South African National Heritage Web Portal built with Metadata Aggregation

Data Provider Interfaces

Alex Priscu†
Department of Computer Science
University of Cape Town
Cape Town Western Province
South Africa
prsale003@myuct.ac.za

ABSTRACT

To preserve historical data, South African memory institutions developed digital heritage archives. However, most digital libraries are separate and unlinked, creating the challenge of searching through numerous archives for resources. Metadata aggregation resolves this by creating a central collection for metadata retrieved from different archives. This paper describes the Data Provider Interfaces component of the metadata aggregation system – HERIPORT – a small-scale prototype of a South African National Heritage Web Portal, using the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). Three OAI-PMH Data Provider Interfaces and three (unqualified) Dublin Core (DC) Converter applications were implemented to map the original metadata to DC and expose the DC metadata to be aggregated by the Harvester component. Each interface provides OAI-PMH request and response services to view the metadata of a repository. Functional and non-functional requirements of the Data Provider Interfaces were evaluated by Unit, Correctness, Compliance, Conformance, Integration and User Evaluation testing. Based on the assessment of the test results, it was concluded that the interfaces could successfully convert metadata to DC and expose harvestable metadata using the OAI-PMH. Finally, the resulting Data Provider Interfaces is a proof of concept that can be integrated and reused in future projects and fill the gap in building a low-cost South African National Heritage Web Portal with metadata aggregation.

KEYWORDS

Metadata Aggregation, Conversion, Data Provider, OAI-PMH.

1. INTRODUCTION

1.1 Background

Online digital repository systems, digital libraries or digital archives are a popular framework for the storage of various forms of digital content [8]. Academic documents and heritage collections are commonly archived in such repositories for the purpose of discovery and preservation [9] and have the advantage of general accessibility during the current period of increasing Web browser use [8]. These systems store and manage digital items and provide users with access to them by providing search and browse services through a Web-based interface [9].

1.2 Problem Statement

Early digital archives, such as the Bleek and Lloyd Collection [12] and early publications scanned by Digital Imaging South Africa (DISA) [13] are both independent and not linked. Cross-archive discovery is a problem that affects digital libraries, as there are a large number of existing digital libraries hosted by different institutions [3].

Over the last ten years, efforts have been made to create a national heritage portal, which allows public access to a unified collection specifically based on heritage documents [3]. The Europeana project [2], which links many European archives, is an exemplary but costly project. The South African National Electronic Theses and Dissertations (NETD) portal [10] is an example of a local system, designed and built at a low cost, but for a different document domain.

The above-mentioned cross-archive discoverability problem can be resolved by an Organisational Architecture such as metadata aggregation, which facilitates the discovery of various cultural heritage resources through the collection of related metadata [3][15]. The most popular technology used for cultural heritage data is the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [2], as it focuses on metadata aggregation and meets unique criteria for cultural heritage metadata aggregation [3]. In order to share repository content with remote Organised Architecture Service Providers using the OAI-PMH, an OAI-compliant Web interface is required to run on the Data Provider's server [6].

1.3 Project Aim

The aim of this project was to create a small-scale prototype of a South African National Heritage Web Portal, using best practices in low-resource heritage archive systems to collect metadata spanning multiple local heritage archives and provide end-users with services such as searching and browsing across the archives using a central Web portal. This paper focuses on the design and implementation of the Data Provider Interfaces component of the prototype as a software engineering project. The aim of the Data Provider Interfaces was to map the original metadata to (unqualified) Dublin Core (DC) [19] and expose the metadata using the OAI-PMH [6].

1.4 High Level System Architecture

A three-layer architecture for the development of HERIPORT was adopted as the Metadata Aggregator System. This architecture enabled the overall system to be divided into three main components that have been developed and deployed as stand-alone projects. Each layer is a component that is separate and communicates through the fixed low-cost OAI-PMH [7]. The system layers shown in Figure 1 are: Data Provider Interfaces, Harvester, and Web Portal. Each Data Provider Interface was required in order to share the metadata of each Data Provider repository. Without these interfaces, metadata would not have reached the Harvester and Web Portal. The architecture of the Data Provider Interface is shown in Section 3.3.2.1.

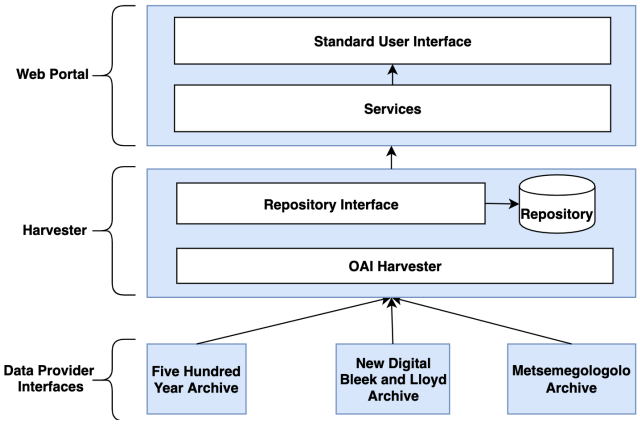

Figure 1: HERIPORT System Architecture Diagram

1.5 Report Overview

This report is composed of six main sections: Related Work; Design and Implementation; Evaluation / Testing; Results and Discussion; Conclusions; and Future Works.

2. RELATED WORK

2.1. Metadata Aggregation

Online digital libraries contain and can share a large number of resources [2]. Potential users face the challenges of discoverability and usage of these resources due to the fact that individual digital libraries are hosted by separate organisations. A standard solution to these challenges is the aggregation of metadata. Metadata aggregation is where a central organisation [3], called an aggregator, helps the discovery and usage of digital library resources that each digital library cannot offer in isolation [2]. This is achieved by an aggregator collecting or "harvesting" metadata from the storage of digital libraries or "repositories".

Internet search engines are unable to retrieve user-requested cultural heritage resources on the basis of metadata combined with World Wide Web hypertext documents [2]. Consequently, the retrieval of cultural heritage resources through search engines was shown to be ineffective. The software solution to this problem of data synchronisation is the low-cost interoperability protocol - the OAI-PMH [2][6] - and therefore, its frequent implementation [4].

2.2. OAI-PMH

The OAI-PMH is a simple and general Hypertext Transfer Protocol (HTTP) - based client-server protocol that facilitates the incremental transfer of metadata between networked systems [11]. This protocol is an approach to interoperability with minimal interaction between remote sites and the central site, which primarily transfers metadata incrementally [7]. Services are provided to users on internally stored and processed data acquired by an OAI-PMH Service Provider from an OAI-PMH Data Provider [15]. By avoiding remote metadata searching and allowing the transfer of metadata, there is less interaction between remote sites in a scattered discovery environment [11].

Typically, there are two types of entities in the OAI-PMH interactions [7]: Data Providers that expose their metadata to interested clients, and Service Providers that provide value-added services based on the metadata collected from Data Providers [4].

Recently, more metadata aggregators have been able to obtain metadata-only information from individual data sources due to an increase in the number of digital repositories worldwide [4]. If a digital library offers an OAI-PMH interface to its repository, a Service Provider may access the repository of the digital library [11]. The digital content of the repository is systematically accessed using less bandwidth and with improved stability of service provision than remote searching of metadata [11].

2.2.1 OAI Repositories

Repositories are accessible network servers of Data Providers that store digital data [6]. The digital data includes items that relate to the metadata records harvested from a Data Provider. If a repository supports a collection of OAI-PMH requests, it is called OAI-compliant [6]. The association between records and items is many-to-one, as the metadata of an item can be expressed in multiple formats.

Specifically intended, the nature of items is indistinct, as the OAI-PMH is defined as unaware of the Data Provider's standards [6]. The OAI-PMH supports Data Providers that have fixed metadata content, those that computationally derive metadata in different formats from some intermediate form or content itself, or those that are stores or intermediaries for external content providers of metadata.

Furthermore, an OAI-PMH interface [6] is essential for scalability as another repository higher up the hierarchy, such as a continental or international repository, is capable of harvesting a national repository such as in the case of the NETD [10], resulting in minimal duplication of records and network traffic.

2.2.2 OAI-PMH Records

The OAI-PMH defines an Extensible Markup language (XML) encoded byte stream as a “record” [7]. A metadata record is made of three parts and serves as a packaging mechanism for harvested metadata of Data Provider items [6]:

2.2.2.1 Header

The first part contains the information required for harvesting, such as a unique record identifier and a datestamp indicating the date the record metadata was created, deleted, or changed. Header information is included in all metadata records [6].

2.2.2.2 Metadata

The second part comprises metadata in a singular format like (unqualified) Dublin Core [7], which is a metadata schema consisting of 15 fields commonly used to describe data [19]. All OAI-PMH Data Providers need to be able to share DC metadata records [6][7].

2.2.2.3 About

The third part is an optional container to store record metadata information. The container could store metadata rights information and conditions for metadata use. The OAI-PMH does not define this container's inner structure [6].

2.2.3 OAI-PMH Requests

Data Providers are repositories in which structured metadata is exposed through an OAI-PMH Web interface [6]. Service Providers send requests for the OAI-PMH service to collect the metadata from Data Provider repositories. The OAI-PMH is defined by a set of six “verbs” or “requests” that are invoked in the HTTP context [7]. This protocol uses the HTTP POST or GET methods [6]. The purpose of these methods is to simplify the configuration of OAI-compliant repositories for Data Providers by using readily available Web tools. All OAI-PMH requests have the following structure of elements [6]:

2.2.3.1 base-URL
Comprises of the Internet hostname and the HTTP server port functioning as a repository [7], optionally, with an OAI-PMH request path specified for the HTTP server [6].

2.2.3.2 keyword arguments
Comprises of a list of key-value pairs [7]. At a minimum, each OAI-PMH request has one key-value pair that specifies the name of one of the six verbs / requests [6]. Arguments can be optional or required, depending on the request. Possible keys include *verb, identifier, metadataPrefix, from, until, set* and *resumptionToken*.

Figure 2 illustrates an example of an OAI-PMH request to the NETD site [10], where the keys are verb and metadataPrefix. The value of verb is ListRecords, and the value of metadataPrefix is oai_dc. The corresponding shortened response to the request is shown in Figure 3.

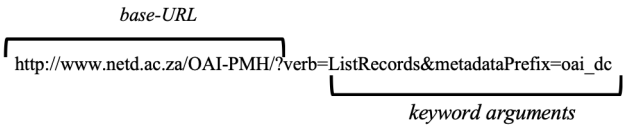

Figure 2: Example OAI-PMH Request to the NETD [10]

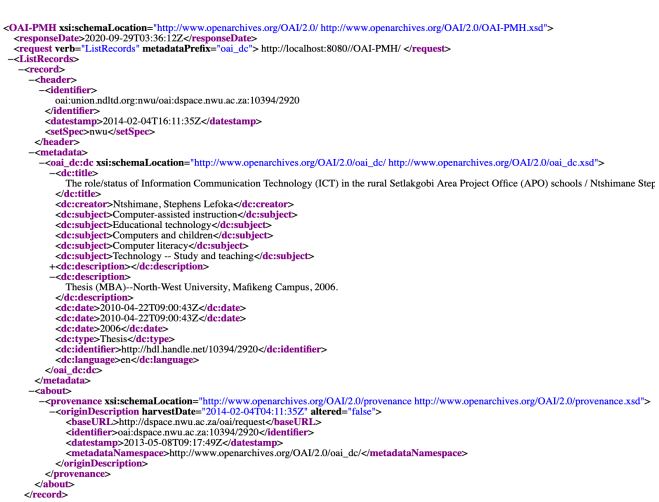

Figure 3: Example OAI-PMH Response from the NETD [10]

Responses to all HTTP requests in an OAI-PMH interface are encoded in XML. Each response contains the OAI-PMH request that generated it, which allows responses to be processed in batch. The OAI-PMH requests are listed in Table 1 with their responses [6].

Table 1: OAI-PMH Requests and Responses

Protocol Request	Response
GetRecord	Returns an individual metadata record from an OAI repository using the required record identifier and metadata prefix arguments.
Identify	Returns general information about an OAI repository such as the repository's name, base-URL, supported OAI-PMH version and the administrator's email address.
ListIdentifiers	Returns metadata record identifiers that can be harvested from an OAI repository using the required metadata prefix argument and optional set association in a repository and date period arguments.
ListMetadataFormats	Returns all available metadata formats in an OAI repository. Optionally using an identifier argument to restrict the request for a particular metadata record and its' available formats.
ListRecords	Returns all metadata records stored in an OAI repository that can be harvested by using the required metadata prefix argument and optional set association and date period arguments.
ListSets	Returns the structural information of the sets available within an OAI repository.

2.2.4 OAI-PMH Data Provider Experiences

The National Science Digital Library (NSDL) is an American metadata aggregator that collects Web-based educational resources from Science, Technology, Engineering and Math (STEM) Data Provider repositories to provide easy access for educators, learners and the general public to teach and learn [16]. The NSDL stores standard DC metadata and NSDL-specific variants, as well as indexed full-text content. The NSDL implements the OAI-PMH and uses a metadata repository to centrally store and manage metadata from multiple locations [17]. It includes a component that normalises metadata to a standard OAI-PMH format. This metadata is then moved to the metadata repository, which exposes the metadata by using the OAI-PMH for harvesting by a Service Provider.

A similar system was developed at the University of Illinois, providing access to cultural heritage resources [1]. The project studied the effectiveness of OAI-PMH harvesting and aggregating metadata. Normalisation was performed to improve metadata record discovery by enforcing consistent content formats. This process included obtaining and analysing field values, learning how Data Providers interpreted their native fields in relation to the DC fields, and implementing a controlled vocabulary. Metadata was then normalised. The normalisation process was found to be difficult to automate due to varying metadata formats from the metadata providers and needed to be manually examined. For example, Type field normalisation was only processed for items that appeared frequently due to the required manual effort [15]. The DC fields Date, Coverage, and Format were the most unproblematic to normalise [1].

3 DESIGN AND IMPLEMENTATION

3.1 Requirements Gathering

3.1.1 Requirements Gathering Method

In order to develop the three Data Provider Interfaces, the requirements of each Data Provider were required. The first iteration of the project began by initialising communication with representatives of the Data Providers: The New Digital Bleek and Lloyd Archive, The Five Hundred Year Archive and The Metsemegologolo Archive. Using email as a means of communication, we introduced the aim of our project and confirmed their support and interest by asking them to complete a set of four questions. Questions involved asking each Data Provider for their archive metadata format, their verbal consent for the use of their metadata, as well as a sample of metadata records. The objective of this task was to obtain metadata from all three Data Providers and to establish a communication channel between the stakeholders and the project team. The procurement of the XML metadata enabled early software development of the Data Provider Interfaces to begin.

Following the first email concerning the provision of data, a second email concerning the specific data requirements of each Data Provider was sent. The responses from the three Data Providers completed the gathering of requirements for the project and Data Provider Interfaces. Each response gave direction to which tags of the archive metadata should be mapped to the resulting 15 DC fields and what ownership rights should be linked with the metadata records.

The chosen method of email as a means of communication with the Data Providers proved to be effective as the response time was short and fitted within our assigned task duration as set out in our project Gantt Chart. The project schedule was planned to minimise task dependencies between each Data Provider, such as the gathering of requirements and sample metadata. The time allocated to the gathering of requirements was estimated on the basis of the initial communication response time. Tasks were made to be

performed concurrently to make optimal use of time. Our method of gathering requirements may attribute its success to the current COVID-19 pandemic, as lockdown regulations resulted to an increased amount of time spent home and online.

3.1.2 Functional Requirements Gathered

From the requirements that were gathered, the Data Provider Interfaces were all required to preserve specific metadata tags when converting to DC. The tags included: Title, Subject, Description, Date, Type, Format, Identifier, Rights, Author and Contributor. If records were missing values for these tags, default values were assigned. The Data Provider Interfaces were also required to provide a datestamp on each converted record so that they could be searched by date when harvesting. The metadata mapping schema is further described in Section 3.3.2.3. The overall requirements of the Data Provider Interfaces were to map the original metadata to (unqualified) Dublin Core (DC) [19] and expose the metadata using the OAI-PMH [6]. This is illustrated in the UML Use Case Diagram of the Data Provider Interface, Figure 13, in Supplementary Information.

3.1.3 Non-functional Requirements Gathered

Availability of metadata was the non-functional requirement gathered from the Data Providers in the requirements gathering task. The Data Providers requested that the sample metadata not be made publicly available as the metadata consists of media / data not belonging to the Data Provider. They have agreements with partner institutions that each contributed content on their own terms and conditions and made the media / data available under the Creative Commons License: CC-BY-NC-ND. Consequently, any provision of data should not renounce any of the agreements in their memoranda of understanding.

3.2 Implementation Approach

3.2.1 Software Development Methodology

In order to meet the requirements that have been gathered, this project adopted the Agile Software Development Methodology as it proved to be more efficient for projects with versatile user requirements and adjustments than the Waterfall approach [5]. This methodology suited this project as three Data Providers were involved. Due to the complexity and separation of the project components, elements of the Agile and Extreme Programming were used to develop the Data Provider Interfaces.

Our implementation strategy followed a four-phased iterative approach where each iteration followed the Agile Analysis, Development, and Testing Development Cycle [5]. The Agile approach to this project helped keep both the software being developed and the development process simple. Agile also ensured that the interfaces worked even if they were not functionally complete. While Agile and Extreme Programming differ in that Agile anticipates changes and Extreme Programming does not design with change in mind, the combination of the two complemented each other. Agile took into account any evolving stakeholder requirements while maintaining a fixed timeframe for each iteration, developing incremental updates for each interface per iteration, completing each Data Provider Interface feature before moving to another, integrating testing throughout the project lifecycle, and communicating iteratively with all stakeholders.

Before each iteration, software deliverables were defined in the form of a Gantt Chart, and weekly Skype meetings with all team members and project supervisor were held to ensure scheduled development. After each iteration, refactoring was completed using the Extreme Programming Framework to refine the source code. Refactoring benefits included increased efficiency, readability, comprehensibility, and reusability, which is important to the research community. Following this iterative approach made it easier to manage time and complexity constraints while developing each interface. Early project iterations mitigated any identified risks and weekly communication kept the project stakeholders up to date with the development of the project.

3.2.2 Version Control

Version control was used to protect the source code of the Data Provider Interfaces from a loss of work catastrophe. By tracking each change to the source code, progress could be monitored over time. Work could be safely stored in a remote repository which enabled the current source code version to be restored to an earlier version. Git's distributed version control system was chosen to enable work to be completed in a local repository and pushed or pulled to various locations. This was useful when working remotely on a laptop, pushing Git updates, and pulling the latest source code to the Ubuntu server hosting the HERIPORT website. This private server was used in order to meet the non-functional requirement of availability.

3.2.3 Implementation Overview

3.2.3.1 Iteration 1: Setup Metadata Conversion

Following from the requirements gathering and procurement of data, various design artefacts were produced that assisted in the development of the Data Provider Interfaces. These included: a Data Provider Architecture Diagram, a Data Provider Interface UML Activity Diagram and three Data Provider Metadata Mapping Schemas. These artefacts are further described in Section 3.3.2.

Once the functional requirements of the Data Providers were established, and sample XML metadata received, it was required that each Data Provider Interface had their native formatted metadata records mapped to DC as the OAI-PMH Web service requires this format in order to allow external access to the metadata. Firstly, The New Digital Bleek and Lloyd Archive DC Converter application was developed. This archive was chosen first as its metadata was received earliest of the three Data Providers. In order to create the metadata converter, the metadata records and tags were first reviewed and then compared to the Data Provider's field mapping requirements. The mappings were then implemented, and Unit tested. After further development of the DC Converter, Conformance testing was performed by creating and running a DC Converter Tester application.

3.2.3.2 Iteration 2: Expose Converted Metadata

In order to share the newly converted metadata with the Harvester, an OAI-PMH Data Provider Interface for The New Digital Bleek and Lloyd Archive was required. Development of the interface began by creating a Common Gateway Interface (CGI) Python script for handling each of the six OAI-PMH verb keyword arguments. Each verb was implemented, and then Unit and Correctness tested. Testing highlighted responses missing field values and records that could not be displayed. After further development of the DC Converters and Data Provider Interface, Unit testing was completed, and Integration testing was performed with the Harvester component.

3.2.3.3 Iteration 3: Adapt to Data Provider

The third iteration of the project focused on adapting the DC Converter and the OAI-PMH Data Provider Interface developed for the The New Digital Bleek and Lloyd Archive to The Five Hundred Year Archive. The same process of development steps completed in Iterations 1 and 2 for The New Digital Bleek and Lloyd Archive were followed.

3.2.3.4 Iteration 4: Complete Interfaces

The fourth and final iteration of the project focused on developing the Data Provider Interface for The Metsemegologolo Archive. The same process of development steps completed in Iterations 1, 2 and 3 for The New Digital Bleek and Lloyd Archive and The Five Hundred Year Archive were followed.

To successfully meet the project requirements of the Data Provider Interfaces being compliant with the OAI-PMH standard and preserve the interoperability property, Compliance testing of the interfaces was then performed. Testing highlighted the absence of an OAI-PMH concept known as a resumptionToken. Implementation of the token for an interface did not require an excess of project time. Although core functionality had to be altered for specific OAI-PMH verbs to list a limit of 10 records at a time, this was simplified due to the refactoring of the code in each iteration and following the Agile Software Development Methodology. Completion of the final iteration resulted in the successful development of three Data Provider Interfaces that are separable and reusable components. Ideally, the interfaces would have run on the servers of The New Digital Bleek and Lloyd Archive, The Five Hundred Year Archive and The Metsemegologolo Archive. However, by storing the interfaces and copies of the metadata on the HERIPORT server, small, incremental iterations could be produced rapidly, and testing could be integrated throughout the project lifecycle.

3.3 Design

3.3.1 Development Environment

This section identifies the software development environment used to develop and test the three Data Provider Interfaces.

3.3.1.1 Programming Language

The Python Programming Language was selected for the development of the Data Provider Interfaces. Python has the ability to be used on a server to create Web applications and to read and modify files, which was part of the interface requirements. It was necessary to run the interfaces on a Web server using CGI. It was found that there is a CGI module in the core library of Python. As a result, simple CGI scripts that met the project requirements of printing an HTTP response and handle inputs from HTML forms or request parameters could be created. The similarity to the implementation of the OAI-PMH made Python the best choice for the Data Provider Interfaces development.

3.3.1.2 Tools and Technologies

XML - was used for two main purposes. It was firstly used for converting Data Provider XML metadata records into DC formatted XML records. The XML formatting was applied within the DC Converters as discussed in Section 3.3.4. Secondly, XML has been used for the OAI-PMH responses as the OAI-PMH standard requires well-formed XML instance document responses encoded in UTF-8 [7]. The required XML formatting was applied within the OAI-PMH Data Provider Interfaces as discussed in Section 3.3.5.

Ubuntu Web server - was used to host the Data Provider Interfaces that received and outputted OAI-PMH requests and responses. Requests are concatenated with the server URL and XML responses are displayed on a Web page.

3.3.1.3 Packages

Imported Python packages assisted in the development of the DC Converters and OAI-PMH Data Provider Interfaces. These included:

xmltodict - enabled DC Converters to convert Data Provider XML metadata records to a more usable dictionary format variable that stored key tags and key-accessible dictionary values.

pprint - enabled DC Converters to perform Unit testing by printing arbitrary Python data structures in a "prettier" format than the standard library.

simpledc - enabled DC Converters to convert original XML metadata records to a DC cleaned dictionary. With **xmlutils** and **lxml**, DC Converters could print the converted XML tree metadata to an XML file and be validated by the DC Converter Tester as described in Section 4.6.1.

datetime - enabled DC Converters to retrieve the conversion date for the default metadata <dc:date> field as well as enable the Data Provider Interfaces to retrieve the date and time in Coordinated Universal Time (UTC) format so it may be printed in an OAI-PMH response.

unicodedata - enabled DC Converters to normalise non-ASCII characters in the original XML metadata records when converting to DC.

os - enabled DC Converters to write converted XML metadata records to file directories, and Data Provider Interfaces to read XML metadata records directory contents for OAI-PMH responses.

cgi - enabled Data Provider Interfaces to be seamlessly implemented as a CGI script for printing OAI-PMH responses to HTTP OAI-PMH requests and reading HTTP arguments to process OAI-PMH requests.

3.3.2 Design of Data Provider Interfaces

Three design artefacts were developed from the requirements gathering performed in Iteration 1.

3.3.2.1 Data Provider Architecture

Figure 4 abstracts the overall Data Provider Interface component design. The Data Provider Interface runs on an Ubuntu Web server and receives OAI-PMH requests as input. The diagram illustrates how the interface is designed to process HTTP requests, parse arguments, create arguments-based record requests, and produce XML record responses. As seen in the diagram, the Service Provider sends an OAI-PMH request to the OAI-PMH Data Provider. The request is then handled by methods located in the Python script. These methods include reading the original metadata and converting it to the DC format in the DC Converter script. The Data Provider Interface script then requests the DC converted metadata and returns the appropriate OAI-PMH response to the Service Provider, who then stores the metadata in their harvested metadata collection.

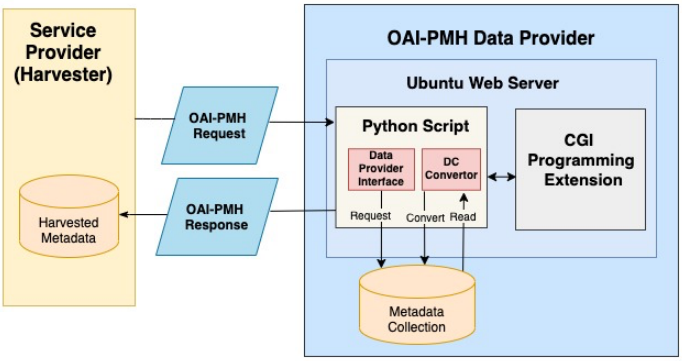

Figure 4: Data Provider Architecture Diagram

3.3.2.2 Data Provider Processes

Figure 5 describes the sequential steps that the Data Provider Interfaces perform when processing an OAI-PMH request till an XML response is generated.

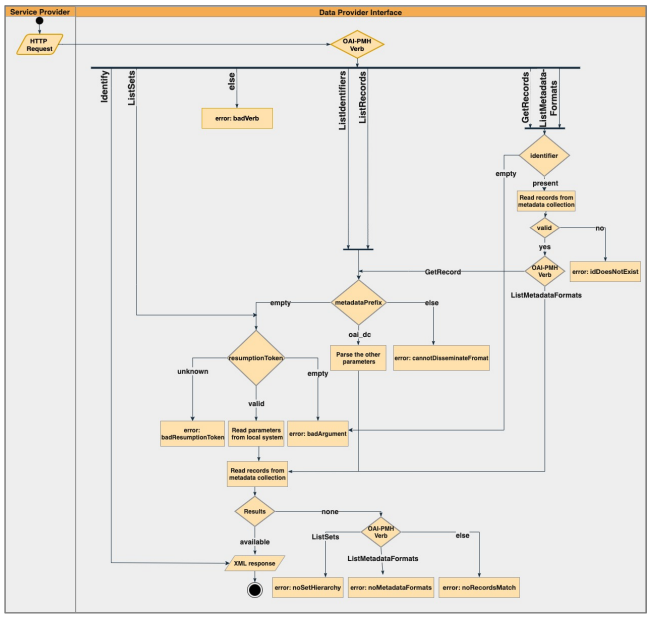

Figure 5: Data Provider Interface UML Activity Diagram

The OAI-PMH verbs shown in Figure 5 were discussed in Table 1 and the resumptionToken is further described in Section 3.3.5. As illustrated in Figure 5, if the request included illegal arguments or the required argument such as metadataPrefix was missing, the badArgument error would be printed. If the value of the resumptionToken argument was invalid or expired the badResumptionToken error would be printed.If the value of the metadataPrefix argument in a request was not supported by the repository the cannotDisseminateFormat error would be printed. If the value of the identifier argument was unknown or illegal in the repository the idDoesNotExist error would be printed. If the combination of the values of the from, until, and set arguments resulted in an empty list the noRecordsMatch error would be printed. If the archive did not support sets the noSetHierarchy error would be printed. If there were no metadata formats available for the specified item noMetadataFormats error would be printed.

3.3.2.3 Data Provider Metadata Mapping Schemas

The design of metadata field mappings was based on the functional requirements of the Data Providers and the available record tags. If no suitable tag was found, a Data Provider-based - default - value was assigned. The mappings for each Data Provider are listed in Table 2. A "None" entry indicates no reasonable value was found.

Table 2: Data Provider Metadata Mapping Schemas

Unqualified DC Field	Digital Bleek and Lloyd Archive Tag	Five Hundred Year Archive Tag	Metsemegologolo Archive Tag
<dc:title>	<title>	<title>	<title>
<dc:subject>	<collection>	<qubitParentSlug>	default <collection>
<dc:description>	<summary>	<file>	default <description>
<dc:creator>	<author>	<scopeAndContent>	<subject>
<dc:publisher>	default <publisher>	<scopeAndContent>	<subject>
<dc:contributor>	default <contributor>	<eventActor>	default <contributor>
<dc:date>	default <date>	default <date>	default <date>
<dc:type>	<type>	<radGeneralMaterial-Designation>	default <type>
<dc:format>	default <format>	default <format>	default <format>
<dc:identifier>	default <identifier>	default <identifier>	default <identifier>
<dc:source>	default <source>	<scopeAndContent>	default <source>
<dc:coverage>	none	none	<provenance>
<dc:rights>	default <rights>	<reproductionConditions>	<rights>

For The New Digital Bleek and Lloyd Archive, default field values were based on the record index number. For example, records with an index value within the range of 1-128 belonged to the Wilhelm Bleek notebooks collection and 134-940 belonged to the Lucy Lloyd |xam notebooks collection. For indexes not in the appropriate ranges, a default value was entered based on the Data Provider name. For example, if the <subject> tag was missing and the record belonged to the Wilhelm Bleek notebooks collection, the value assigned to the <dc:subject> field would be "Wilhelm Bleek notebooks", otherwise if it did not belong to any existing collection the value assigned would be "The New Digital Bleek and Lloyd Archive."

3.3.4 DC Converter Design

Three DC Converter applications were designed and developed based on the Data Provider metadata mapping schemas design artefact shown in Section 3.3.2.3. Python was used to develop each of the application scripts named: BleekAndLloydDCConverter.py, FHYADCConverter.py and MetsemegologoDCConverter.py. All three scripts are designed to convert XML metadata records read in from a directory path and output the converted DC XML metadata records in the same directory so that they can be accessed by the OAI-PMH Data Provider Interfaces.

Sample records from the Data Providers adhered to an incremental number indexing convention for each directory sub-folder. As a result, each sub-folder could be accessed by the corresponding number and each sub-folder contained an XML metadata file. For example, the "stories" directory in The New Digital Bleek and Lloyd Archive, contained 2057 individual sub-folders. The DC identifier field was mapped to the Data Provider Internet hostname ("http://pumbaa.cs.uct.ac.za/~balnew/metadata/stories/") concatenated with the number of the original metadata's sub-folder (1-2057). This allowed the identifier value to be a valid Uniform Resource Identifier (URI) as a user may click on this value and be directed to the original item on The New Digital Bleek and Lloyd Archive website.

The differences in the three Data Provider DC Converters are how the applications filter data type values and metadata formats using conditional statements, and which tags are mapped to corresponding DC fields. Once a file was converted the terminal printed "Successfully created file: metadata-#-dc.xml". When all files were converted, the terminal printed "Successfully converted files." Figure 6 illustrates a native formatted XML metadata record, and Figure 7 illustrates the metadata record after running FHYADCConverter.py.

```
<item>
   <qubitParentSlug>fhya-depot</qubitParentSlug>
   <title>Trade and politics in Southern Mozambique and Zululand in the eighteenth and early nineteenth centuries</title>
   <radGeneralMaterialDesignation>Textual record</radGeneralMaterialDesignation>
   <levelOfDescription>item</levelOfDescription>
   <scopeAndContent>[Source - Benathi Marufu for FHYA, 2020: A copy of David William Hedges&apos; Doctoral thesis titled Trade and politics in S
   Mozambique and Zululand in the eighteenth and early nineteenth centuries. School of Oriental and African Studies University of London 1978.]</
   scopeAndContent>
   <reproductionConditions>Creative Commons License: CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/3.0/

Unless otherwise stated the copyright of all material on the FHYA resides with the contributing institution/custodian.</reproductionConditions>
   <event>
      <eventActor id="internal307">Five Hundred Year Archive</eventActor>
      <eventType>Online curation</eventType>
      <eventDate>2020 -</eventDate>
      <eventDescription>A digital copy from the School of Oriental and African Studies University of London</eventDescription>
   </event>
   <event>
      <eventActor id="internal320">David. W. Hedges</eventActor>
      <eventType>Writing -</eventType>
      <eventDate>1978</eventDate>
      <eventDescription></eventDescription>
   </event>
   <view>
      <title>Trade and politics in Southern Mozambique and Zululand in the eighteenth and early nineteenth centuries</title>
      <file>FHYA%20Depot/Hedges_Thesis.pdf</file>
   </view>
</item>
```

Figure 6: FHYA Depot / 9 / metadata.xml

```
<?xml version='1.0' encoding='utf-8'?>
<oai_dc:dc xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/
oai_dc/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openarchives.or
/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
   <dc:contributor>Five Hundred Year Archive</dc:contributor>
   <dc:contributor>David. W. Hedges</dc:contributor>
   <dc:creator>Benathi Marufu</dc:creator>
   <dc:date>2020-10-01</dc:date>
   <dc:description>http://emandulo.apc.uct.ac.za/collection/FHYA%20Depot/Hedges_Thesis.pdf</dc:description>
   <dc:format>text/xml</dc:format>
   <dc:identifier>http://emandulo.apc.uct.ac.za/metadata/FHYA Depot/9</dc:identifier>
   <dc:publisher>Benathi Marufu</dc:publisher>
   <dc:rights>Creative Commons License: CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/3.0/

Unless otherwise stated the copyright of all material on the FHYA resides with the contributing institution/
custodian.</dc:rights>
   <dc:source>[Source - Benathi Marufu for FHYA, 2020: A copy of David William Hedges' Doctoral thesis titled
   Trade and politics in Southern Mozambique and Zululand in the eighteenth and early nineteenth centuries.
   School of Oriental and African Studies University of London 1978.]</dc:source>
   <dc:subject>fhya-depot</dc:subject>
   <dc:title>Trade and politics in Southern Mozambique and Zululand in the eighteenth and early nineteenth
   centuries</dc:title>
   <dc:type>Textual record</dc:type>
</oai_dc:dc>
```

Figure 7: FHYA Depot / 9 / metadata-9-dc.xml

3.3.5 OAI-PMH Data Provider Interface Design

Three OAI-PMH Data Provider Interfaces have been designed and developed based on the architecture and process design artefacts shown in Section 3.3.2. Python was used to develop each of the interface scripts named: BleekAndLloydOAIInterface.py, FHYAOAIInterface.py and MetsemegogoloOAIInterface.py. All three scripts were designed to receive OAI-PMH requests and output OAI-PMH responses based on DC metadata records generated by the DC Converters. This is illustrated in the UML Sequence Diagram of the Data Provider Interface, Figure 14, in Supplementary Information.

Using conditional statements, each script responded to OAI-PMH verb values. For each of the verbs, keyword arguments were validated before requesting any metadata and generating a response. After implementing argument test cases, the XML response for each of the verbs was defined. The GetRecord verb was implemented by taking in the required identifier and metadataPrefix values. The interface would then scan the records for the corresponding Data Provider set directory. Using the identifier and metadataPrefix values, the verb would then find and validate the record with these values. If the record was found, the program would print the result to an output in the OAI-PMH standard. Otherwise, a standard OAI-PMH error message would be printed. Metadata records included their set repository membership in ListIdentifiers, ListRecords, and GetRecord response headers for selective harvesting. Records also included an optional provenance statement in the about container, giving insight into its provenance. Figure 8 illustrates an OAI-PMH request to The Five Hundred Year Archive interface, and Figure 9 illustrates the OAI-PMH response.

```
http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=GetRecord&identifier=http://emandulo.apc.uct.ac.za/collection/metadata/FHYA%20Depot/6&metadataPrefix=oai_dc
```

Figure 8: FHYAOAIInterface.py OAI-PMH Request

```
<OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2020-10-01 16:24:22.420857</responseDate>
 -<request verb="GetRecord" identifier="http://emandulo.apc.uct.ac.za/collection/metadata/FHYA Depot/6" metadataPrefix="oai_dc">
    http://emandulo.apc.uct.ac.za/metadata/FHYA%20Depot/
  </request>
 -<GetRecord>
  -<record>
    -<header>
      -<identifier>
          http://emandulo.apc.uct.ac.za/collection/metadata/FHYA Depot/6
        </identifier>
        <datestamp>2020-10-01 16:24:22.420881</datestamp>
        <setSpec>FHYA Depot</setSpec>
      </header>
    -<metadata>
      -<oai_dc:dc xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
          <dc:contributor>Five Hundred Year Archive (FHYA)</dc:contributor>
          <dc:contributor>Nafisa Essop Sheik</dc:contributor>
          <dc:creator>Benathi Marufu</dc:creator>
          <dc:date>2020-10-01</dc:date>
        -<dc:description>
            http://emandulo.apc.uct.ac.za/collection/FHYA%20Depot/Sheik_thesis.pdf
          </dc:description>
          <dc:format>text/xml</dc:format>
        -<dc:identifier>
            http://emandulo.apc.uct.ac.za/metadata/FHYA Depot/6
          </dc:identifier>
          <dc:publisher>Benathi Marufu</dc:publisher>
        -<dc:rights>
            Creative Commons License: CC BY-NC-ND https://creativecommons.org/licenses/by-nc-nd/3.0/ Unless otherwise stated the copy
            FHYA resides with the contributing institution/custodian.
          </dc:rights>
        -<dc:source>
            [Source - Benathi Marufu for FHYA, 2020: A copy of Nafisa Essop Sheik's doctoral thesis titled Colonial Rites: Custom, Marriag
            difference in Natal, 1830s - c.1910.]
          </dc:source>
          <dc:subject>fhya-depot</dc:subject>
        -<dc:title>
            Colonial Rites: Custom, Marriage Law and the Making of difference in Natal, 1830s - c.1910
          </dc:title>
          <dc:type>Textual record</dc:type>
        </oai_dc:dc>
      </metadata>
    </record>
  </GetRecord>
</OAI-PMH>
```

Figure 9: FHYAOAIInterface.py OAI-PMH Response

For the OAI-PMH interfaces, a resumptionToken was implemented as ListRecords and ListIdentifiers responses deal with large collections from Data Providers. This essentially allowed a Data Provider Interface to limit the amount of records returned as a result to a request. In a request, a token was used to gain more parts of an incomplete response list. Without a resumptionToken, the interfaces would generate massive XML responses that would cause both Data Providers and Service Providers problems.

Interface requests that did not specify a resumptionToken would receive a response with a list of the first ten records. If the list was incomplete, at the end of the response, a resumptionToken would be displayed as an XML tag. If an argument such as metadataPrefix was entered in a request with resumptionToken, the badResumptionToken error would be printed, as resumptionToken is an exclusive argument. Tokens also enabled repeating requests within their 10-minute lifespan using their value. The token's next record identifier value informed a Harvester where to harvest next. An empty token value indicated the completion of the response list.

Tokens were formatted into five fields. The first field contained the value of the next record identifier after entering the current token. The second field contained the expiration date. The third field contained the metadataPrefix value from the original request. The fourth, fifth and sixth fields were optional. If these values were missing in the original request, default values were assigned but not displayed. These three fields contained the value of arguments from, until, and set. For example, if ListRecords was requested without a token and the from and until keyword arguments were specified and set not, the token would contain the value for from and until, but not set. Optionally, tokens included their expiration datestamp and estimated complete list size. Figure 9 shows this token format in an example response to a request specifying only the required metadataPrefix argument.

```
<resumptionToken expirationDate="2020-09-29T15:54:20" completeListSize="2057">11,e2020-09-29T15:54:20,oai_dc</resumptionToken>
```

Figure 9: Example resumptionToken Format

3.4 Development Challenges

Multiple challenges were faced during the development of three Data Provider Interfaces, with varying project impact factors. The project proposal identified most of these challenges where we identified, analysed and ranked each risk. A risk monitoring, mitigation and management plan

was then developed as a risk matrix. The project's most likely challenge was that its life-cycle development occurred throughout the COVID-19 global pandemic, with no expected outcome to end soon. This challenge mostly affected communication with project stakeholders and team members. Despite this consequence being averted by digital communication, the undecidability of which methods to choose to conduct User Evaluations due to the pandemic still hindered in-person User Evaluation testing and feedback on the interfaces. To reduce the impact of this challenge, we planned various test methods under all possible circumstances, such as Unit, Correctness, Conformance and Compliance Testing, and selected the most appropriate method. Regional load shedding was another environmental challenge that affected the Web development of the interfaces. Due to the load shedding scheduling, we could plan project development to work around load shedding hours. This occasionally resulted in Web-based interfaces not being tested, but this did not affect the timeline of the project.

Each Data Provider Interface required a specific set of test metadata. At the start of the project, the provision of metadata by the three Data Providers was delayed. In order to avoid delaying project tasks requiring metadata, such as the conversion of metadata to comply with an OAI-PMH Data Provider Interface, software development began with the sample metadata provided by the project supervisor. In the early project iterations, the team used a virtual machine called Amazon EC2 Linux, which Amazon calls an “instance”. Due to computing capacity costs in the Amazon Web Services (AWS) cloud, the team opted for a cost-free solution. This virtual machine choice provided insufficient access speed, slowing project development. Our project supervisor, hosting an Ubuntu server at University of Cape Town (UCT) with higher computing capacity, helped overcome this challenge.

4. EVALUATION/TESTING

The development and implementation of the Data Provider Interfaces were evaluated using five test categories. Iterative testing was performed in each of the four project iterations.

4.1 Unit Testing

Unit testing was implemented during project development to validate each Data Provider software code performed as expected. During the development of the DC Converter in Iteration 1, Unit testing of individual metadata records was performed to evaluate whether the DC Converter mapped individual metadata record tags to the expected DC fields and determine if the functional requirement of preserving specific metadata tags when converting to DC was met. During the development of the OAI-PMH Data Provider Interface in Iteration 2, Unit testing was performed to first evaluate whether the interface generated responses to each of the six OAI-PMH request verbs and bad verb values. Unit testing of keyword argument values was then implemented to validate the date format for the Data Provider Interfaces’ from and until keys. Unit test cases were then implemented for the fields entered in the keyword arguments such as testing for illegal arguments, repeating arguments and values containing illegal syntax. Depending on the selected verb, some arguments were required and some optional.

4.2 Correctness Testing

During the development of the OAI-PMH Data Provider Interfaces in Iterations 2, 3 and 4, Correctness testing was performed to validate each OAI-PMH verb XML response format including error cases such as badArgument, badResumptionToken, cannotDisseminateFormat, idDoesNotExist, noRecordsMatch, noSetHierarchy and noMetadataFormats by observing and comparing the output to the OAI-PMH standard website [7].

4.3 Conformance and Compliance Testing

After the development of the first DC Converter in Iteration 1, a DC Conformance testing application (DCConverterTester.py) was developed to evaluate whether the DC Converters could produce XML metadata that conformed to the DC schema [18]. This framework is further described in Section 4.6.1. After the development of all the OAI-PMH Data Provider Interfaces in Iteration 4, a software Repository Explorer tool developed by Hussein Suleman [14] was chosen to evaluate whether the interfaces complied with the OAI-PMH standard [7]. This tool is further described in Section 4.6.2.

4.4 Integration Testing

After the development of each OAI-PMH Data Provider Interface in Iterations 2, 3 and 4, Integration testing was carried out with the Harvester to determine whether the Data Provider Interfaces could interact successfully with the Harvester component of the HERIPORT metadata aggregator system. Using the set of six OAI-PMH verbs, each test case evaluated whether the Harvester can make a valid OAI-PMH request to each of the three Data Provider Interfaces and receive a standard OAI-PMH response.

4.5 User Evaluation Testing

Data Provider Interface Usability testing was carried out in Iteration 4. Questions and tasks were asked to determine whether the functional requirements gathered had been fulfilled. A total of 10 participants performed the evaluation.

4.6 Testing Frameworks

4.6.1 DC Converter Tester Application

This testing framework was implemented as a Python script and run from the command line. The input was a valid name for one of the three DC Converters. The Tester then ran the DC Converter and performed two tests on each DC Data Provider metadata record. The first case tested if XML was well-formed. The second case tested metadata validity against the imported DC schema file - oai_dc.xsd. This schema is illustrated in Figure 15, in Supplementary Information. After each record was read in and tested, the total number of successfully converted records was compared to the total number of records read in. Figure 10 illustrates input of the MetsemegologoloDCConverter.py to the Tester application, and the Tester application output.

```
_____________________________________________________________________
                • Unqualified Dublin Core Conversion Tester •
_____________________________________________________________________
Enter the name of the Unqualified Dublin Core Conversion Script: MetsemegologoloDCConverter.py
_____________________________________________________________________
Importing: MetsemegologoloDCConverter.py
Running: MetsemegologoloDCConverter.py
Successfully created file: metadata-1-dc.xml
Successfully converted files.
Testing: MetsemegologoloDCConverter.py
_____________________________________________________________________


Testing: Metsemegologolo/1/metadata-1-dc.xml
Case 1: Checking XML Well-Formedness
PASS

Case 2: Validate against Unqualified Dublin Core Schema
PASS
_____________________________________________________________________
Report: 1 out of 1 files converted successfully.
```

Figure 10: DCConverterTester.py Example

4.6.2 OAI-PMH Repository Explorer

This testing tool is an offline package compiled and run from the command line. The tool command included the execution command, the HERIPORT server URL with the Data Provider Interface to-test directory, and the language chosen to display the results. Figure 11 shows this format using the HERIPORT server's Internet hostname and the path to the Interface, and Figure 12 shows the output of the tool. A total of 42

tests were performed to validate the OAI-PMH standard response [7] of each Data Provider Interface.

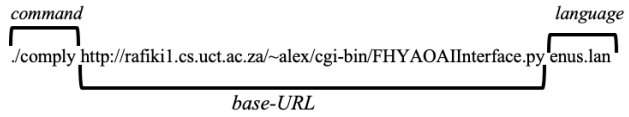

Figure 11: OAI-PMH Repository Explorer Example Input

```
(1) Testing : Identify
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=Identify
Test Result : OK
---- [ Repository Name = The Five Hundred Year Archive ]
---- [ Protocol Version = 2.0 ]
---- [ Base URL = http://emandulo.apc.uct.ac.za/metadata/FHYA%20Depot/ ]
---- [ Admin Email = prsale003@myuct.ac.za ]
---- [ Granularity = YYYY-MM-DD ]
---- [ Earliest Datestamp = 2020-09-20 ]

(2) Testing : Identify (illegal_parameter)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=Identify&test=test
Test Result : OK
```

Figure 12: OAI-PMH Repository Explorer Example Output

5. RESULTS AND DISCUSSION

5.1 Unit Testing

Table 3: Final Unit Testing Results

Testing	Digital Bleek and Lloyd Archive	Five Hundred Year Archive	Metsemegologolo Archive
DC Converter	**PASS**	**PASS**	**PASS**
Tag to DC Field Mapping	PASS	PASS	PASS
Empty tags	PASS	PASS	PASS
Non-ASCII values	PASS	PASS	PASS
OAI-PMH Interface	**PASS**	**PASS**	**PASS**
Six verbs and bad verb	PASS	PASS	PASS
from and until arguments	PASS	PASS	PASS
Illegal arguments	PASS	PASS	PASS
Repeating arguments	PASS	PASS	PASS
Illegal syntax	PASS	PASS	PASS

From Unit testing, it was found that some DC fields of the metadata records contained no values in the OAI-PMH responses and specific metadata records could not be displayed in an OAI-PMH response as they contained non-ASCII characters. Unit testing was then performed in Iteration 2 on the DC Converters to evaluate whether they successfully normalised non-ASCII characters after further development. Table 3 shows that all Unit test cases of the DC Converters and OAI-PMH Interfaces passed.

5.2 Conformance Testing of DC Converter Applications

Table 4: Conformance Testing Results

Data Provider	Case 1 Result	Case 2 Result	Successful Files	Total files	Conversion Success rate
Digital Bleek and Lloyd Archive	PASS	PASS	87	87	100%
Five Hundred Year Archive	PASS	PASS	2057	2057	100%
Metsemegologolo Archive	PASS	PASS	1	1	100%

Results from the Conformance testing of DC Converters in Table 4 show that The New Digital Bleek and Lloyd Archive DC Converter achieved 2057 successful conversions out of 2057 records equating to a 100% success rate, The Five Hundred Year Archive DC Converter achieved 87 successful conversions out of 87 records equating to a 100% and The Metsemegologolo Archive DC Converter achieved one successful conversion out of one record, equating to a 100% success rate. These results indicate that all records were successfully converted to the DC standard and each DC Converter performed correctly.

5.3 Compliance Testing of Data Provider Interfaces

Table 5: Final Compliance Testing Results

Test Case ID	Testing	Digital Bleek and Lloyd Archive	Five Hundred Year Archive	Metsemegologolo Archive
1	Identify	PASS	PASS	PASS
2	Identify (illegal_parameter)	PASS	PASS	PASS
3	ListMetadataFormats	PASS	PASS	PASS
4	ListSets	PASS	PASS	PASS
5	ListSets (resumptionToken)	SKIPPED	SKIPPED	SKIPPED
6	ListIdentifiers (oai_dc)	PASS	PASS	PASS
7	ListIdentifiers (resumptionToken)	PASS	PASS	SKIPPED
8	ListIdentifiers (resumptionToken, oai_dc)	PASS	PASS	SKIPPED
9	ListIdentifiers (oai_dc, from/until)	PASS	PASS	PASS
10	ListIdentifiers (oai_dc, set, from/until)	SKIPPED	SKIPPED	SKIPPED
11	ListIdentifiers (oai_dc, illegal_set, illegal_from/until)	PASS	PASS	PASS
12	ListIdentifiers (oai_dc, from granularity != until granularity)	PASS	PASS	PASS
13	ListIdentifiers (oai_dc, from > until)	PASS	PASS	PASS
14	ListIdentifiers ()	PASS	PASS	PASS
15	ListIdentifiers (metadataPrefix)	SKIPPED	SKIPPED	SKIPPED
16	ListIdentifiers (illegal_mdp)	PASS	PASS	PASS
17	ListIdentifiers (mdp, mdp)	PASS	PASS	PASS
18	ListIdentifiers (illegal_resumptiontoken)	PASS	PASS	PASS
19	ListIdentifiers (oai_dc, from YYYY-MM-DD)	PASS	PASS	PASS
20	ListIdentifiers (oai_dc, from YYYY-MM-DDThh:mm:ssZ)	PASS	PASS	PASS
21	ListIdentifiers (oai_dc, from YYYY)	PASS	PASS	PASS
22	ListMetadataFormats (identifier)	PASS	PASS	PASS
23	ListMetadataFormats (illegal_id)	PASS	PASS	PASS
24	GetRecord (identifier, metadataPrefix)	SKIPPED	SKIPPED	SKIPPED
25	GetRecord (identifier, oai_dc)	PASS	PASS	PASS
26	GetRecord (identifier)	PASS	PASS	PASS
27	GetRecord (identifier, illegal_mdp)	PASS	PASS	PASS
28	GetRecord (oai_dc)	PASS	PASS	PASS
29	GetRecord (illegal_id, oai_dc)	PASS	PASS	PASS
30	GetRecord (invalid_id, oai_dc)	PASS	PASS	PASS
31	ListRecords (oai_dc, from/until)	PASS	PASS	PASS
32	ListRecords (resumptionToken)	SKIPPED	SKIPPED	SKIPPED
33	ListRecords (metadataPrefix, from/until)	SKIPPED	SKIPPED	SKIPPED
34	ListRecords (oai_dc, illegal_set, illegal_from/until)	PASS	PASS	PASS
35	ListRecords	PASS	PASS	PASS
36	ListRecords (oai_dc, from granularity != until granularity)	PASS	PASS	PASS
37	ListRecords (oai_dc, until before earliestDatestamp)	PASS	PASS	PASS
38	ListRecords (oai_dc, until before earliestDatestamp)	PASS	PASS	PASS
39	ListRecords (illegal_resumptiontoken)	PASS	PASS	PASS
40	ListIdentifiers (oai_dc, set)	SKIPPED	SKIPPED	SKIPPED
41	GetRecord (identifier, oai_dc)	SKIPPED	SKIPPED	SKIPPED
42	IllegalVerb	PASS	PASS	PASS

After Compliance testing, two tests failed for each interface. These cases highlighted the absence of a resumptionToken. Once the resumptionToken development was completed and version controlled, all three Data Provider Interfaces were tested again for Compliance and Integration. Results from the Compliance testing of the interfaces are shown in Table 5, where cases were either passed, failed or skipped. Cases were skipped if they couldn't be completed based on the collection of metadata or if the case should not have been performed due to prior cases. The table shows that no cases failed, indicating that all OAI-PMH requests returned

expected responses for each of the six verbs, and that the Data Provider Interfaces for each Data Provider produced valid OAI-PMH standard responses.

5.4 Integration Testing of Data Provider Interfaces

Table 6: Final Integration Testing Results

Test Case ID	Description	Digital Bleek and Lloyd Archive	Five Hundred Year Archive	Metsemegologolo Archive
1	Harvester can invoke GetRecord request and receive a response.	PASS	PASS	PASS
2	Harvester can invoke Identify request and receive a response	PASS	PASS	PASS
3	Harvester can invoke ListIdentifiers request and receive a response	PASS	PASS	PASS
4	Harvester can invoke ListMetadataFormats request and receive a response	PASS	PASS	PASS
5	Harvester can invoke ListRecords request and receive a response	PASS	PASS	PASS
6	Harvester can invoke ListSets request and receive a response	PASS	PASS	PASS

Results from the Integration testing of Data Provider Interfaces with the Harvester in Table 6 show that the Harvester could successfully request and receive expected responses for each of the six OAI-PMH verbs. These results of no failure cases indicate the Data Provider Interfaces can be harvested from, in a metadata aggregator system.

5.5 User Evaluation Test of Data Provider Interfaces

Table 7: User Evaluation Test Results

Test Case ID	Question	Digital Bleek and Lloyd Archive		Five Hundred Year Archive		Metsemegologolo Archive	Students				
Participant No.		1	2	3	4	5	6	7	8	9	10
1	Do you see the metadata for this specific field?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
2	Did the metadata records display fields you were expecting?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
3	Did the metadata records display fields about the correct item?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
4	Did the identifier field link to the original metadata item?	Y	Y	Y	Y	N	Y	Y	Y	Y	Y

Results from the User Evaluation test with 5 Data Provider and 5 student participants are shown in Table 7, where "Y" stands for "Yes" and "N" for "No". The table shows that 100 percent of participants were able to see the metadata for a specific field, that the expected fields were displayed for metadata records, and that the metadata records displayed fields about the correct item. 90 Percent of participants said the identifier field linked to the original metadata item. The one participant who answered "No" was the only representative from The Metsemegologolo Archive, as this Data Provider does not have an active website hosting the item. These results indicate that Data Provider Interfaces fulfilled the functional requirements gathered in Iteration 1.

6. CONCLUSIONS

This software engineering project led to the creation of three OAI-PMH Data Provider Interfaces and three DC Converter applications that expose metadata to be harvested in the HERIPORT metadata aggregator system. The Data Provider Interfaces provide OAI-PMH request and response services to view the metadata of a repository. The DC Converters provided the required metadata to the interfaces by converting the metadata records to DC. The Data Provider Interfaces passed Unit, Correctness, Compliance, Conformance, Integration and User Evaluation testing and met the functional and non-functional requirements. Based on the assessment of the test results, it was concluded that Data Provider Interfaces could successfully convert the metadata to DC and expose the metadata using the OAI-PMH. A limitation of the Data Providers' metadata is the inclusion of offensive language. This could be improved by spending more time cleaning up the metadata of Data Providers. The limitation of not making the metadata available to the public limited Data Provider Interfaces to not be run on the public Data Providers' servers, and only the private HERIPORT server. It was also concluded that this software component prototype was a successful proof of concept in building the Data Provider Interfaces layer of a three-layer metadata aggregator system architecture. Overall, the Data Provider Interfaces can be integrated and reused in future projects and fill the gap in building a low-cost South African National Heritage Web Portal with metadata aggregation.

7. FUTURE WORKS

For future work, Data Provider Interfaces could be extended by implementing a graphical user interface to map metadata. This interface would run before the metadata repository is exposed via an OAI-PMH Web interface, allowing the Data Provider to choose the metadata format they want to convert to. The Data Provider could also choose which tags to map to fields of the chosen metadata format, where to store the converted metadata, and what the naming convention would be. This interface could be multilingual to support international Data Providers, whereas the current interface only supports English. This future work could be a valuable tool for the global research community, increasing the availability of metadata harvesting in all domains.

8. ACKNOWLEDGEMENTS

I would like to acknowledge Hussein Suleman's contribution as project supervisor, creator of the NETD Web Portal and the OAI Repository Tool Explorer. I would also like to thank the three Data Providers, The New Digital Bleek and Lloyd Archive, The Five Hundred Year Archive and The Metsemegologolo Archive, for their participation in this project.

REFERENCES

[1] Timothy W. Cole, 2003. Implementation of a Scholarly Information Portal Using the Open Archives Initiative Protocol for Metadata Harvesting.

[2] Nuno Freire, Antoine Isaac, Glen Robson, John Howard, and Hugo Manguinhas, 2017. A survey of Web technology for metadata aggregation in cultural heritage. *Information Services & Use,* Vol. 37, Issue 4, 425–436.

[3] Nuno Freire, Rene Voorburg, Roland Cornelissen, Sjors de Valk, Enno Meijers, E and Antoine Isaac, 2019. Aggregation of Linked Data in the Cultural Heritage Domain: A Case Study in the Europeana Network, 252.

[4] Nikos Houssos, Kostas Stamatis, Vangelis Banos, Sarantos Kapidakis, Emmanouel Garoufallou, Alexandros Koulouris, 2011. Implementing enhanced OAI-PMH requirements for Europeana. Research and Advanced Technology for Digital Libraries, 396-407.

[5] Ajay Kumar, Pavan Kumar, Naga Malleswari, Sathvika, 2018. A Study on SDLC For Water Fall and Agile. International Journal of Engineering & Technology 7, 2.32 (2018), 10.

[6] Carl Lagoze and Herbet Van de Sompel, 2001. The Open Archives Initiative. *In Proceedings of the 1st Joint Conference on Digital Libraries (JCDL'2001) (Roanoke, Va.).* 54–62.

[7] Carl Lagoze, Herbert Van de Sompel, Michael Nelson, and Simon Warner, 2002. The Open Archives Initiative Protocol for Metadata Harvesting, Open Archives Initiative. Available http://www.openarchives.org/OAI/openarchivesprotocol.html

[8] Hussein Suleman, Marc Bowes, Matthew Hirst and Suraj Subrun. Hybrid online-offline digital collections, 2010. *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on - SAICSIT '10, ACM Press*, 421-425.

[9] Hussein Suleman, 2019. Reflections on Design Principles for a Digital Repository in a Low Resource Environment. *Proceedings of HistoInformatics Workshop 2019, 13 September 2019, Oslo, Norway, CEUR.*

[10] Hussein Suleman, Tatenda Chipeperekwa and Lawrence Webley, 2011. Creating a National Electronic Thesis and Dissertation Portal in South Africa, in Olivier, E., and Suleman, H. (eds): Proceedings of 14th International Symposium on Electronic Theses and Dissertations (ETD 2011), Cape Town, 13- 15 September. Available http://dl.cs.uct.ac.za/conferences/etd2011/papers/etd2011_webley.pdf

[11] Hussein Suleman, Edward A Fox, 2002. Towards Universal Accessibility of ETDs: Building the NDLTD Union Archive, *Proceedings of The Fifth International Symposium on Electronic Theses and Dissertations (ETD 2002), Provo, Utah, USA*, Available at http://www.wvu.edu/~thesis/proceedings.html

[12] Hussein Suleman, 2007. In-Browser Digital Library Services, *Proceedings of Research and Advanced Technology for Digital Libraries, 11th European Conference (ECDL 2007), 16-19 September 2007,* Budapest, Hungary, 462-465.

[13] Hussein Suleman, 2008. An African Perspective on Digital Preservation, *in Post-Proceedings of International Workshop on Digital Preservation of Heritage and Research Issues in Archiving and Retrieval, Kolkata, India, 29-31 October 2007.* Available http://www.husseinspace.com/research/publications/iwdph_2007_african.pdf

[14] Hussein Suleman, 2019. Repository Explorer | digital libraries laboratory @ uct. cs. Dl.cs.uct.ac.za. http://dl.cs.uct.ac.za/projects/re/.

[15] Sarah Shreeves, Joanne Kaczmarek, Timothy Cole, 2003. Harvesting cultural heritage metadata using the OAI protocol. Library High Tech 21, 2, 159-169

[16] Hussein Suleman, 2012. Design and architecture of digital libraries. in Chowdhury, G.G. and Foo, S. (Eds), *Digital Libraries and Information Access: Research Perspectives,* Facet, London, pp. 13-28.

[17] Herbert Van de Sompel and Carl Lagoze, 2002. Notes from the interoperability front: A progress report on the Open Archives Initiative. *In Proceedings of the Sixth European Conference on Digital Libraries,* ed. Maristella Agnosti and Constantiano Thanos, 144–157. Rome: Springer. http://www.openarchives.org/ documents/ecdl-oai.pdf

[18] Simeon Warner, 2004. Open Archives Initiative Protocol for Metadata Harvesting. *http://www.openarchives.org/OAI/2.0/oai_dc.xsd*.

[19] Stuart Weibel, 2005. The Dublin Core: A Simple Content Description Model for Electronic Resources. *Bulletin of the American Society for Information Science and Technology*. 24, 1 (2005), 9-11.

SUPPLEMENTARY INFORMATION

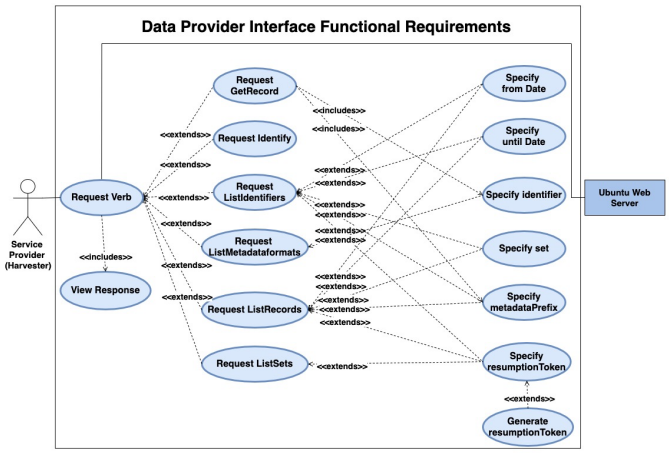

Figure 13: UML Use Case Diagram of the Data Provider Interface

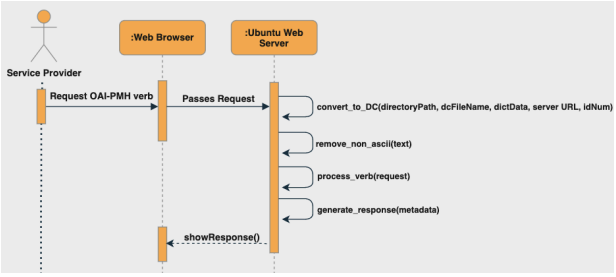

Figure 14: UML Use Sequence Diagram of the Data Provider Interface

```
<schema xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/" xmlns:dc="http://purl.org/dc/
elements/1.1/" xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.openarchives.org/
OAI/2.0/oai_dc/" elementFormDefault="qualified" attributeFormDefault="unqualified">
<annotation>
<documentation> XML Schema 2002-03-18 by Pete Johnston. Adjusted for usage in the OAI-PMH. Schema
imports the Dublin Core elements from the DCMI schema for unqualified Dublin Core. 2002-12-19 updated to
use simpledc20021212.xsd (instead of simpledc20020312.xsd) </documentation>
</annotation>
<import namespace="http://purl.org/dc/elements/1.1/" schemaLocation="http://dublincore.org/schemas/xmls/
simpledc20021212.xsd"/>
<element name="dc" type="oai_dc:oai_dcType"/>
<complexType name="oai_dcType">
<choice minOccurs="0" maxOccurs="unbounded">
<element ref="dc:title"/>
<element ref="dc:creator"/>
<element ref="dc:subject"/>
<element ref="dc:description"/>
<element ref="dc:publisher"/>
<element ref="dc:contributor"/>
<element ref="dc:date"/>
<element ref="dc:type"/>
<element ref="dc:format"/>
<element ref="dc:identifier"/>
<element ref="dc:source"/>
<element ref="dc:language"/>
<element ref="dc:relation"/>
<element ref="dc:coverage"/>
<element ref="dc:rights"/>
</choice>
</complexType>
</schema>
```

Figure 15: oai_dc.xsd

```
Alexs-MacBook-Pro:Repository_Explorer-2.0-1.47 AlexPriscu$ ./comply http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/BleekAndLloydOAIInterface.py enus.lan
Open Archives Initiative :: Protocol for Metadata Harvesting v2.0
RE Protocol Tester 1.47 :: UCT AIM :: September 2014

(1) Testing : Identify
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=Identify
Test Result : OK
---- [ Repository Name = The New Digital Bleek and Lloyd ]
---- [ Protocol Version = 2.0 ]
---- [ Base URL = http://pumbaa.cs.uct.ac.za/~balnew/metadata/stories/ ]
---- [ Admin Email = prsale003@myuct.ac.za ]
---- [ Granularity = YYYY-MM-DD ]
---- [ Earliest Datestamp = 2020-10-01 ]

(2) Testing : Identify (illegal_parameter)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=Identify&test=test
Test Result : OK

(3) Testing : ListMetadataFormats
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListMetadataFormats
Test Result : OK
---- [ Only oai_dc supported ]

(4) Testing : ListSets
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListSets
Test Result : OK

(5) Skipping : ListSets (resumptionToken)
This test is being skipped because it cannot or should not be performed.

(6) Testing : ListIdentifiers (oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc
Test Result : OK
---- [ Sample Identifier = http://pumbaa.cs.uct.ac.za/~balnew/metadata/stories/1 ]
---- [ Identifier Resumption Token = 11,e2020-10-02T11:39:58,oai_dc ]

(7) Testing : ListIdentifiers (resumptionToken)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&resumptionToken=11,e2020-10-
02T11:39:58,oai_dc
Test Result : OK

(8) Testing : ListIdentifiers (resumptionToken, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&resumptionToken=11,e2020-10-
02T11:39:58,oai_dc&metadataPrefix=oai_dc
Test Result : OK

(9) Testing : ListIdentifiers (oai_dc, from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2000-01-
01&until=2000-01-01
Test Result : OK

(10) Skipping : ListIdentifiers (oai_dc, set, from/until)
This test is being skipped because it cannot or should not be performed.

(11) Testing : ListIdentifiers (oai_dc, illegal_set, illegal_from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&set=really_wrong_set&from=some_random_date&until=some_random
_date
Test Result : OK

(12) Testing : ListIdentifiers (oai_dc, from granularity != until granularity)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001-01-
01&until=2002-01-01T00:00:00Z
Test Result : OK

(13) Testing : ListIdentifiers (oai_dc, from > until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2000-01-
01&until=1999-01-01
Test Result : OK

(14) Testing : ListIdentifiers ()
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers
Test Result : OK

(15) Skipping : ListIdentifiers (metadataPrefix)
This test is being skipped because it cannot or should not be performed.

(16) Testing : ListIdentifiers (illegal_mdp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=illegal_mdp
Test Result : OK

(17) Testing : ListIdentifiers (mdp, mdp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&metadataPrefix=oai_dc
Test Result : OK

(18) Testing : ListIdentifiers (illegal_resumptiontoken)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&resumptionToken=junktoken
Test Result : OK

(19) Testing : ListIdentifiers (oai_dc, from YYYY-MM-DD)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001-01-01
Test Result : OK

(20) Testing : ListIdentifiers (oai_dc, from YYYY-MM-DDThh:mm:ssZ)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001-01-
01T00:00:00Z
Test Result : OK

(21) Testing : ListIdentifiers (oai_dc, from YYYY)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001
Test Result : OK

(22) Testing : ListMetadataFormats (identifier)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/BleekAndLloydOAIInterface.py?verb=ListMetadataFormats&identifier=http://pumbaa.cs.uct.ac.za/~balnew/metadata/stories/1
Test Result : OK
---- [ Only oai_dc supported ]

(23) Testing : ListMetadataFormats (illegal_id)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListMetadataFormats&identifier=really_wrong_id
Test Result : OK

(24) Skipping : GetRecord (identifier, metadataPrefix)
This test is being skipped because it cannot or should not be performed.

(25) Testing : GetRecord (identifier, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/BleekAndLloydOAIInterface.py?verb=GetRecord&identifier=http://pumbaa.cs.uct.ac.za/~balnew/metadata/stories/1&metadataPrefix=oai_dc
Test Result : OK

(26) Testing : GetRecord (identifier)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/BleekAndLloydOAIInterface.py?verb=GetRecord&identifier=http://pumbaa.cs.uct.ac.za/~balnew/metadata/stories/1
Test Result : OK

(27) Testing : GetRecord (identifier, illegal_mdp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/BleekAndLloydOAIInterface.py?verb=GetRecord&identifier=http://pumbaa.cs.uct.ac.za/~balnew/metadata/stories/1&metadataPrefix=really_w
rong_mdp
Test Result : OK

(28) Testing : GetRecord (oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=GetRecord&metadataPrefix=oai_dc
Test Result : OK

(29) Testing : GetRecord (illegal_id, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/BleekAndLloydOAIInterface.py?verb=GetRecord&identifier=really_wrong_id&metadataPrefix=oai_dc
Test Result : OK

(30) Testing : GetRecord (invalid_id, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=GetRecord&identifier=invalid\"id&metadataPrefix=oai_dc
Test Result : OK

(31) Testing : ListRecords (oai_dc, from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&from=2000-01-
01&until=2000-01-01
Test Result : OK

(32) Skipping : ListRecords (resumptionToken)
This test is being skipped because it cannot or should not be performed.

(33) Skipping : ListRecords (metadataPrefix, from/until)
This test is being skipped because it cannot or should not be performed.

(34) Testing : ListRecords (oai_dc, illegal_set, illegal_from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/BleekAndLloydOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&set=really_wrong_set&from=some_random_date&until=some_random_dat
e
Test Result : OK

(35) Testing : ListRecords
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListRecords
Test Result : OK

(36) Testing : ListRecords (oai_dc, from granularity != until granularity)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&from=2001-01-
01&until=2002-01-01T00:00:00Z
Test Result : OK

(37) Testing : ListRecords (oai_dc, until before earliestDatestamp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&until=2019-10-01
Test Result : OK

(38) Testing : ListRecords (oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc
Test Result : OK

(39) Testing : ListRecords (illegal_resumptiontoken)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=ListRecords&resumptionToken=junktoken
Test Result : OK

(40) Skipping : ListIdentifiers (oai_dc, set)
This test is being skipped because it cannot or should not be performed.

(41) Skipping : GetRecord (identifier, oai_dc)
This test is being skipped because it cannot or should not be performed.

(42) Testing : IllegalVerb
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/BleekAndLloydOAIInterface.py?verb=IllegalVerb
Test Result : OK


---- Total Errors : 0


Alexs-MacBook-Pro:Repository_Explorer-2.0-1.47 AlexPriscu$ ./comply http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py
enus.lan
Open Archives Initiative :: Protocol for Metadata Harvesting v2.0
RE Protocol Tester 1.47 :: UCT AIM :: September 2014

(1) Testing : Identify
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=Identify
Test Result : OK
---- [ Repository Name = The Five Hundred Year Archive ]
---- [ Protocol Version = 2.0 ]
---- [ Base URL = http://emandulo.apc.uct.ac.za/metadata/FHYA%20Depot/ ]
---- [ Admin Email = prsale003@myuct.ac.za ]
---- [ Granularity = YYYY-MM-DD ]
---- [ Earliest Datestamp = 2020-10-01 ]

(2) Testing : Identify (illegal_parameter)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=Identify&test=test
Test Result : OK

(3) Testing : ListMetadataFormats
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListMetadataFormats
Test Result : OK
---- [ Only oai_dc supported ]

(4) Testing : ListSets
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListSets
Test Result : OK

(5) Skipping : ListSets (resumptionToken)
This test is being skipped because it cannot or should not be performed.

(6) Testing : ListIdentifiers (oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc
Test Result : OK
---- [ Sample Identifier = http://emandulo.apc.uct.ac.za/metadata/FHYA Depot/1 ]
---- [ Identifier Resumption Token = 11,e2020-10-02T11:38:17,oai_dc ]

(7) Testing : ListIdentifiers (resumptionToken)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&resumptionToken=11,e2020-10-02T11:38:17,oai_dc
Test Result : OK

(8) Testing : ListIdentifiers (resumptionToken, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&resumptionToken=11,e2020-10-
02T11:38:17,oai_dc&metadataPrefix=oai_dc
Test Result : OK

(9) Testing : ListIdentifiers (oai_dc, from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2000-01-
01&until=2000-01-01
Test Result : OK

(10) Skipping : ListIdentifiers (oai_dc, set, from/until)
This test is being skipped because it cannot or should not be performed.

(11) Testing : ListIdentifiers (oai_dc, illegal_set, illegal_from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&set=really_wrong_set&from=some_random_date&until=some_random_date
Test Result : OK

(12) Testing : ListIdentifiers (oai_dc, from granularity != until granularity)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001-01-
01&until=2002-01-01T00:00:00Z
Test Result : OK

(13) Testing : ListIdentifiers (oai_dc, from > until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2000-01-
01&until=1999-01-01
Test Result : OK

(14) Testing : ListIdentifiers ()
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers
Test Result : OK

(15) Skipping : ListIdentifiers (metadataPrefix)
This test is being skipped because it cannot or should not be performed.

(16) Testing : ListIdentifiers (illegal_mdp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=illegal_mdp
Test Result : OK

(17) Testing : ListIdentifiers (mdp, mdp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&metadataPrefix=oai_dc
Test Result : OK

(18) Testing : ListIdentifiers (illegal_resumptiontoken)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&resumptionToken=junktoken
Test Result : OK

(19) Testing : ListIdentifiers (oai_dc, from YYYY-MM-DD)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001-01-01
Test Result : OK

(20) Testing : ListIdentifiers (oai_dc, from YYYY-MM-DDThh:mm:ssZ)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001-01-01T00:00:00Z
Test Result : OK

(21) Testing : ListIdentifiers (oai_dc, from YYYY)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001
Test Result : OK

(22) Testing : ListMetadataFormats (identifier)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/FHYAOAIInterface.py?verb=ListMetadataFormats&identifier=http://emandulo.apc.uct.ac.za/metadata/FHYA Depot/1
Test Result : OK
---- [ Only oai_dc supported ]

(23) Testing : ListMetadataFormats (illegal_id)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListMetadataFormats&identifier=really_wrong_id
Test Result : OK
```

```
(24) Skipping : GetRecord (identifier, metadataPrefix)
This test is being skipped because it cannot or should not be performed.

(25) Testing : GetRecord (identifier, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/FHYAOAIInterface.py?verb=GetRecord&identifier=http://emandulo.apc.uct.ac.za/metadata/FHYA Depot/1&metadataPrefix=oai_dc
Test Result : OK

(26) Testing : GetRecord (identifier)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/FHYAOAIInterface.py?verb=GetRecord&identifier=http://emandulo.apc.uct.ac.za/metadata/FHYA Depot/1
Test Result : OK

(27) Testing : GetRecord (identifier, illegal_mdp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/FHYAOAIInterface.py?verb=GetRecord&identifier=http://emandulo.apc.uct.ac.za/metadata/FHYA Depot/1&metadataPrefix=really_wrong_mdp
Test Result : OK

(28) Testing : GetRecord (oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=GetRecord&metadataPrefix=oai_dc
Test Result : OK

(29) Testing : GetRecord (illegal_id, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=GetRecord&identifier=really_wrong_id&metadataPrefix=oai_dc
Test Result : OK

(30) Testing : GetRecord (invalid_id, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=GetRecord&identifier=invalid\"id&metadataPrefix=oai_dc
Test Result : OK

(31) Testing : ListRecords (oai_dc, from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&from=2000-01-01&until=2000-
01-01
Test Result : OK

(32) Skipping : ListRecords (resumptionToken)
This test is being skipped because it cannot or should not be performed.

(33) Skipping : ListRecords (metadataPrefix, from/until)
This test is being skipped because it cannot or should not be performed.

(34) Testing : ListRecords (oai_dc, illegal_set, illegal_from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/FHYAOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&set=really_wrong_set&from=some_random_date&until=some_random_date
Test Result : OK

(35) Testing : ListRecords
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListRecords
Test Result : OK

(36) Testing : ListRecords (oai_dc, from granularity != until granularity)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&from=2001-01-01&until=2002-
01-01T00:00:00Z
Test Result : OK

(37) Testing : ListRecords (oai_dc, until before earliestDatestamp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&until=2019-10-01
Test Result : OK

(38) Testing : ListRecords (oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc
Test Result : OK

(39) Testing : ListRecords (illegal_resumptiontoken)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=ListRecords&resumptionToken=junktoken
Test Result : OK

(40) Skipping : ListIdentifiers (oai_dc, set)
This test is being skipped because it cannot or should not be performed.

(41) Skipping : GetRecord (identifier, oai_dc)
This test is being skipped because it cannot or should not be performed.

(42) Testing : IllegalVerb
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/FHYAOAIInterface.py?verb=IllegalVerb
Test Result : OK


---- Total Errors : 0


Alexs-MacBook-Pro:Repository_Explorer-2.0-1.47 AlexPriscu$ ./comply http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py enus.lan
Open Archives Initiative :: Protocol for Metadata Harvesting v2.0
RE Protocol Tester 1.47 :: UCT AIM :: September 2014

(1) Testing : Identify
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=Identify
Test Result : OK
---- [ Repository Name = Metsemegologolo Archive ]
---- [ Protocol Version = 2.0 ]
---- [ Base URL = http://Metsemegologolo.apc.uct.ac.za/metadata/Metsemegologolo/ ]
---- [ Admin Email = prsale003@myuct.ac.za ]
---- [ Granularity = YYYY-MM-DD ]
---- [ Earliest Datestamp = 2020-10-01 ]

(2) Testing : Identify (illegal_parameter)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=Identify&test=test
Test Result : OK

(3) Testing : ListMetadataFormats
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListMetadataFormats
Test Result : OK
---- [ Only oai_dc supported ]

(4) Testing : ListSets
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListSets
Test Result : OK

(5) Skipping : ListSets (resumptionToken)
This test is being skipped because it cannot or should not be performed.

(6) Testing : ListIdentifiers (oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc
Test Result : OK
---- [ Sample Identifier = http://Metsemegologolo.apc.uct.ac.za/metadata/Metsemegologolo/1 ]

(7) Skipping : ListIdentifiers (resumptionToken)
This test is being skipped because it cannot or should not be performed.

(8) Skipping : ListIdentifiers (resumptionToken, oai_dc)
This test is being skipped because it cannot or should not be performed.

(9) Testing : ListIdentifiers (oai_dc, from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2000-01-
01&until=2000-01-01
Test Result : OK

(10) Skipping : ListIdentifiers (oai_dc, set, from/until)
This test is being skipped because it cannot or should not be performed.

(11) Testing : ListIdentifiers (oai_dc, illegal_set, illegal_from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&set=really_wrong_set&from=some_random_date&until=some_rand
om_date
Test Result : OK

(12) Testing : ListIdentifiers (oai_dc, from granularity != until granularity)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001-01-
01&until=2002-01-01T00:00:00Z
Test Result : OK

(13) Testing : ListIdentifiers (oai_dc, from > until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2000-01-
01&until=1999-01-01
Test Result : OK

(14) Testing : ListIdentifiers ()
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers
Test Result : OK

(15) Skipping : ListIdentifiers (metadataPrefix)
This test is being skipped because it cannot or should not be performed.

(16) Testing : ListIdentifiers (illegal_mdp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=illegal_mdp
Test Result : OK

(17) Testing : ListIdentifiers (mdp, mdp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&metadataPrefix=oai_dc
Test Result : OK

(18) Testing : ListIdentifiers (illegal_resumptiontoken)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&resumptionToken=junktoken
Test Result : OK

(19) Testing : ListIdentifiers (oai_dc, from YYYY-MM-DD)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001-01-
01
Test Result : OK

(20) Testing : ListIdentifiers (oai_dc, from YYYY-MM-DDThh:mm:ssZ)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001-01-
01T00:00:00Z
Test Result : OK

(21) Testing : ListIdentifiers (oai_dc, from YYYY)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2001
Test Result : OK

(22) Testing : ListMetadataFormats (identifier)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py?verb=ListMetadataFormats&identifier=http://Metsemegologolo.apc.uct.ac.za/metadata/Metsemegologolo/1
Test Result : OK
---- [ Only oai_dc supported ]

(23) Testing : ListMetadataFormats (illegal_id)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListMetadataFormats&identifier=really_wrong_id
Test Result : OK
```

```
(24) Skipping : GetRecord (identifier, metadataPrefix)
This test is being skipped because it cannot or should not be performed.

(25) Testing : GetRecord (identifier, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py?verb=GetRecord&identifier=http://Metsemegologolo.apc.uct.ac.za/metadata/Metsemegologolo/1&metadataPre
fix=oai_dc
Test Result : OK

(26) Testing : GetRecord (identifier)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py?verb=GetRecord&identifier=http://Metsemegologolo.apc.uct.ac.za/metadata/Metsemegologolo/1
Test Result : OK

(27) Testing : GetRecord (identifier, illegal_mdp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py?verb=GetRecord&identifier=http://Metsemegologolo.apc.uct.ac.za/metadata/Metsemegologolo/1&metadataPre
fix=really_wrong_mdp
Test Result : OK

(28) Testing : GetRecord (oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=GetRecord&metadataPrefix=oai_dc
Test Result : OK

(29) Testing : GetRecord (illegal_id, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py?verb=GetRecord&identifier=really_wrong_id&metadataPrefix=oai_dc
Test Result : OK

(30) Testing : GetRecord (invalid_id, oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py?verb=GetRecord&identifier=invalid\"id&metadataPrefix=oai_dc
Test Result : OK

(31) Testing : ListRecords (oai_dc, from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&from=2000-01-
01&until=2000-01-01
Test Result : OK

(32) Skipping : ListRecords (resumptionToken)
This test is being skipped because it cannot or should not be performed.

(33) Skipping : ListRecords (metadataPrefix, from/until)
This test is being skipped because it cannot or should not be performed.

(34) Testing : ListRecords (oai_dc, illegal_set, illegal_from/until)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-
bin/MetsemegologoloOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&set=really_wrong_set&from=some_random_date&until=some_random_d
ate
Test Result : OK

(35) Testing : ListRecords
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListRecords
Test Result : OK

(36) Testing : ListRecords (oai_dc, from granularity != until granularity)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&from=2001-01-
01&until=2002-01-01T00:00:00Z
Test Result : OK

(37) Testing : ListRecords (oai_dc, until before earliestDatestamp)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc&until=2019-10-01
Test Result : OK

(38) Testing : ListRecords (oai_dc)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListRecords&metadataPrefix=oai_dc
Test Result : OK

(39) Testing : ListRecords (illegal_resumptiontoken)
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=ListRecords&resumptionToken=junktoken
Test Result : OK

(40) Skipping : ListIdentifiers (oai_dc, set)
This test is being skipped because it cannot or should not be performed.

(41) Skipping : GetRecord (identifier, oai_dc)
This test is being skipped because it cannot or should not be performed.

(42) Testing : IllegalVerb
URL : http://rafiki1.cs.uct.ac.za/~alex/cgi-bin/MetsemegologoloOAIInterface.py?verb=IllegalVerb
Test Result : OK


---- Total Errors : 0
```

Figure 16: OAI-PMH Repository Explorer Results