



UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE



COMPUTER SCIENCE HONOURS

FINAL PAPER

2015

Title: Transforming DSpace into a Research Information Management System: Ingestion Manager and Report Writer Components

Author: Darryl Meyer

Project Abbreviation: NRFDB

Supervisor: Assoc. Prof. Hussein Suleman

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

Transforming DSpace into a Research Information Management System: Ingestion Manager and Report Writer Components

Darryl Meyer

Dept. of Computer Science

University of Cape Town

Private Bag X3,

Rondebosch, 7701

South Africa

myrdar003@myuct.ac.za

ABSTRACT

Two databases of the South African National Research Foundation are to be migrated from a legacy database system to a more modern DSpace repository. Opportunities for transforming DSpace into a research information management system were identified, which in turn led to the development of add-ons for DSpace. The add-ons were developed to assist users with tasks that are not currently supported in DSpace. These tasks are: allowing non-admin users to remotely ingest batches of new items and generating detailed reports on the information in a DSpace repository. The add-ons were designed to be simple and require minimal user interaction. The design resulted in add-ons that met all the functional requirements, had acceptable performance for increasing task size, and scored highly in usability tests.

Categories and Subject Descriptors

H.3.7 [Information Storage and Retrieval]: Digital Libraries – Collection, System issues, User issues.

General Terms

Documentation, Design, Human Factors, Verification.

Keywords

DSpace, Research Information Management Systems, Ingestion Management, Report Writing

1. INTRODUCTION

The South African National Research Foundation has identified a need to migrate their existing Current and Completed Research Projects and Funded Projects databases to DSpace. The data in these databases is of particular importance for the retention and promotion of research in South Africa. These databases are to be migrated to a more modern DSpace repository, which will help streamline the ingestion of information from universities across South Africa into the NRF's databases. There are also opportunities to build advanced features into DSpace that will further assist the ingestion of information into these, and other databases.

For online supplementary material, please go to: pubs.cs.uct.ac.za and search for this project by its abbreviation-NRFDB.

This project has identified some of the opportunities as developing add-ons for DSpace that would help transform it into more of a research information management system (RIMS). The add-ons that were developed for this project will assist the users of DSpace to perform tasks that are labour-intensive or would otherwise not be possible. The first add-on is a workflow-based ingestion management system to assist non-admin users of DSpace to import batches of items into DSpace while maintaining the approval workflow system present in DSpace. The second add-on is a report generation tool that provides reporting on items in DSpace that are more detailed than the statistics reports already available in DSpace. Some of the existing approaches to these opportunities require technically skilled users to perform sometimes tedious tasks in order to achieve the same results that these add-ons achieve with minimal user interaction.

This paper presents a review on some of the literature that is relevant to the functionality of the add-ons. In addition, it describes the design and implementation of the add-ons and concludes with discussions on the results of testing the add-ons.

2. BACKGROUND

2.1 Institutional Repositories

A digital repository is a collection of digital objects. It differs from other digital collections in that the architecture of the repository enables management of the content and related metadata [7]. An institutional repository (IR) is a kind of digital repository. An IR is specialised to preserve and disseminate the intellectual output of the researchers at universities or other research institutions [2].

2.2 DSpace

DSpace is an open source IR system that supports the capture of digital works, the distribution of those works over the Internet through a search and retrieval system, and ultimately the long-term preservation of those works [21]. DSpace was developed by a collaboration between Massachusetts Institute of Technology (MIT) Libraries and Hewlett-Packard Labs [15].

2.2.1 DSpace Submission Workflow

Items in DSpace are sorted in a hierarchy of communities, sub-communities and collections [15]. Collections in DSpace allow for the separation of different types or topics of content within a community. DSpace allows for different roles within the system, such as “submitters” and “reviewers.” Separation of roles is useful when the users who submit content are different from those who review and approve the content [6]. DSpace is the first open source solution to tackle the complex problem of allowing different

submission workflows for different communities. Separate workflows help when different communities have different restrictions in place.

2.2.2 Importing into DSpace

There are multiple ways to import data into DSpace. One of these ways, as The Ohio State University did, was to create custom scripts to format legacy data so that it can be imported using DSpace's command line import tool [19]. The legacy data to be imported is usually in the format of Comma Separated Value (CSV) files or spreadsheet documents. These files have to be converted to XML or JSON formatted metadata files, with the metadata in the Dublin Core¹ format, before they can be imported into DSpace.

2.2.3 DSpace REST API

A recent inclusion into the functionality of DSpace is a RESTful web services API [13]. The REST API allows third-party application developers access to the objects in DSpace. First introduced in DSpace version 4, with read-only access, the REST API was only able to provide publicly accessible objects [13]. In DSpace version 5, the REST API allows for user authorization to access restricted content and to view, create, edit and delete objects in DSpace [14].

2.2.4 DSpace Storage Architecture

DSpace was designed to allow for customization and enhancement. Its design is a familiar architecture—the layered architecture. This architecture is separated into three layers with each layer consisting of different components. The three layers are: storage, business logic and application [18]. The storage layer is designed to use the file system to store files and a relational database to store and manage metadata. DSpace can be set up to use either a PostgreSQL or Oracle database [15].

2.3 Research Information Management Systems

A research information management system (RIMS), also known as a current research information system (CRIS), is a system that is used to manage the information about the research process. The information it manages can include: funding opportunities, proposal submissions, bids for funding and active research projects. The fundamental idea of a CRIS is to be able to manage the events of the research process and promote the research outputs [10].

2.3.1 CRIS solutions

CRIS solutions attempt to improve the visibility of published research and to present the results of research in way that it can be easily absorbed [9]. One such CRIS solution is InfoEd Global. InfoEd Global provides solutions for institutions to manage their research produced. Their solutions include managing research from the initial concept to the final publication of the research [8]. The DSpace CRIS module is a good open source alternative to this, and other, established solutions.

2.3.2 DSpace CRIS

Provided with the DSpace source code is an additional module that adds CRIS features to DSpace. The module was developed by Cineca² for the University of Hong Kong and has since been made available as an open source additional module for DSpace. The DSpace CRIS module extends upon the data model of DSpace to better showcase the works and promote the reputations of

researchers, as well as to support the interactions between researchers of different institutions [12].

2.4 Business Intelligence Reporting

Business Intelligence (BI) systems enable reporting and analysis on business operations [20]. Meaningful reports can help managers to make informed strategic decisions. Some BI systems connect directly to data sources to enable generation of reports. DSpace does not include a reporting tool that can match the functionality of a BI solution. It can, however, generate usage statistics such as item submission and view counts [11].

2.4.1 Existing BI Solutions

Some existing open source BI solutions include OpenReports³ and ReportServer⁴. In both these solutions, reports are designed in a separate report designing software and are imported for later use. The development of OpenReports has ended and as a result its functionality is being surpassed by more modern solutions like ReportServer. Furthermore, because ReportServer is under current development, there is an active user community and support from the creators.

2.4.2 Integration into DSpace

Integrating a BI solution into DSpace is made possible because DSpace uses a familiar relational database management system (PostgreSQL or Oracle) to manage metadata. In order to implement a BI solution to work with DSpace, it has to be able to have direct access to DSpace's database. This is accomplished by configuring the BI solution to connect to the database using the correct JDBC driver and connection settings. However, the information in the database has to be interpreted and report templates must be designed before the BI solution can meet any reporting needs.

3. DESIGN

3.1 Requirements Analysis

3.1.1 National Research Foundation

The requirements specified by our client, the South African National Research Foundation, was to migrate the data on two legacy database systems from the STAR Classic Application database to a more modern DSpace repository. The two databases are the Current and Completed Research Projects and the National Research Foundation's Funded Projects databases.

3.1.2 Identified RIMS Components

Further requirements for our project were identified by our project supervisor, Assoc. Prof. Hussein Suleman, and agreed upon by our client. The requirements were to help transform an institutional repository (IR), like DSpace, into more of a research information management system (RIMS). The transformation was done by identifying components of a RIMS that DSpace does not include by default and those were built in as add-ons to DSpace. The components identified were: an automatic metadata mapping tool, a manual metadata mapping tool, a report generation tool, and a workflow-based ingestion management system.

The automatic metadata mapping tool takes in legacy data (a CSV file) as input and attempts to automatically determine the appropriate Dublin Core field type of each column of data in the input. The manual metadata mapper allows the user to manually set the field type for each column, should the automatic metadata mapper have made a mistake. The automatic and manual metadata

¹ <http://dublincore.org/>

² <http://www.cineca.it/en>

³ <http://oreports.com/>

⁴ <http://reportserver.net/en/>

mapping tools were developed by my team member, Craig Feldman.

The report generation tool is able to generate detailed reports on the items stored in a DSpace repository. DSpace provides limited reporting functionality by default but the report generation tool is able to go beyond that and generate detailed reports using the metadata of items, communities and collections in DSpace. The workflow-based ingestion management system supports batch imports of items into DSpace from legacy data inputs. Batch imports are currently supported in DSpace, but only by administrators. This system allows non-administrator users to remotely ingest batches of items into a DSpace repository whilst still maintaining the approval workflow steps as in DSpace. The tool for report generation is called the Report Writer and the tool for the workflow-based ingestion management system is called the Ingestion Manager.

3.1.3 User Survey

A survey was compiled to gather feedback from the DSpace user community about our proposed add-ons. The survey included eight questions designed to gain an understanding on whether our proposed functionality would be useful to the DSpace community and to help refine our initial requirements. The survey was created using Lime Survey⁵ and a link to the survey with a brief description of our project was sent to the IRTalk, DSpace technical support and developers mailing lists. The survey link was also shown after a presentation of our project to the members attending the Digital Libraries workshop at the IFLA IT Section 2015 pre-conference satellite meeting.

The survey was completed by 14 participants and after analysis of the results we could conclude that the functionality we had proposed would be useful to the DSpace community and was worth proceeding with. The results and statistics of the survey can be found in the online supplementary materials.

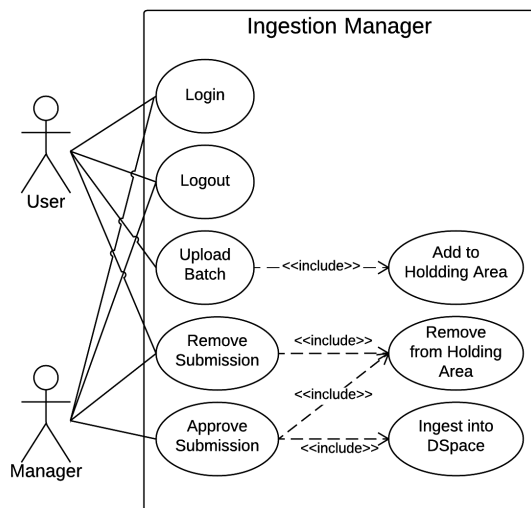


Figure 1. Use Case Diagram for the Ingestion Manager

3.1.4 Use Cases

Use case diagrams and narratives for the Ingestion Manager and Report Writer were drawn up to define the goals of the users of both systems. Once the goals of the users were identified, we began to implement the functionality. The use case narratives for the

Ingestion Manager and Report Writer can be found in the online supplementary materials.

The goals of the users of the Ingestion Manager, as depicted in Figure 1, are separated by the roles of the users. The Ingestion Manager includes two roles: a user and a manager. Both users and managers are able to log in and out of the system. A user can upload a batch and that batch will be entered into the holding area. Both users and managers can remove batches which will remove the batch from the holding area as well. A manager can approve a batch that is in the holding area, which will enter the batch into the DSpace repository and remove the batch from the holding area.

There is only one user of the Report Writer, a manager, as depicted in Figure 2. A manager of the Ingestion Manager and a manager of the Report Writer can be the same person but does not necessarily have to be. A manager of the Report Writer can login, logout, generate reports, and download generated reports.

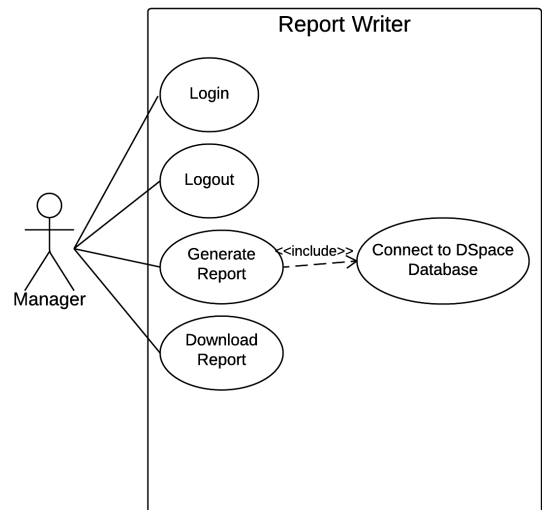


Figure 2. Use Case Diagram for the Report Writer

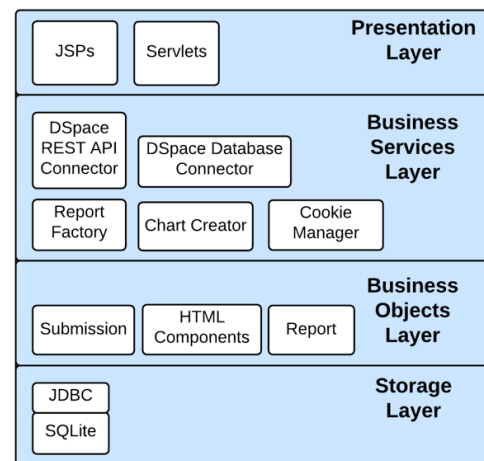


Figure 3. Layered architecture of the applications

3.2 System Architecture and Design

3.2.1 Architecture

A layered architecture model is used in both the Ingestion Manager and the Report Writer. There are 4 layers in the model: storage,

⁵ <https://www.limesurvey.org/en/>

business services, business objects, and presentation. The layers and components of each layer of the architecture is depicted in the architecture diagram—Figure 3.

3.2.2 User Roles

Users of the system are separated into two categories: users and managers. This separation is important to enable a workflow whereby a manager has to approve a user's submission before the submission can be entered into the DSpace repository. A user is able to upload a batch, view or remove batches pending approval, and view batches that have been approved. In the Ingestion Manager, a manager is able to review, remove or approve batches that are pending approval. In the Report Writer, a manager is able to generate reports from the available templates. To keep the reports confidential, only managers are able to access functionality of the Report Writer.

3.2.3 Login

DSpace requires a user to be logged in before they can submit items to the repository. The login functionality of the Ingestion Manager and Report Writer are the same. Upon loading the applications, a user is presented with a login screen. The login screen prompts the user to enter their email address and password so that they can be authenticated in the DSpace repository. Login is handled by interfacing with the REST API of DSpace. A successful call (a user with valid login credentials for the DSpace repository) to the DSpace REST API login endpoint will return a token. The token is stored and later passed to the submission REST API endpoint, together with the item. To log the user out, the token is passed to the logout REST API endpoint. The token is stored in a cookie to maintain the user's logged-in status while they are interacting with the application.

Separate from the DSpace login, the applications also keep a stored record of whether the user attempting to login is a manager. This record is used to redirect the user to the functionality appropriate for their role. The user's login credentials are not stored by the applications; instead only a relation that states if the user has manager permissions is stored.

3.2.4 Ingestion Manger

The Ingestion Manager implements a submission workflow-based system that allows batches of items to be ingested into a DSpace repository. The core functionality of the submission workflow (described in more detail in Subsection 3.2.4.2) is to allow a submission to be approved by a manager before it is ingested into the DSpace repository. Other functions of the Ingestion Manager were built to add more control and convenience to the core functionality.

The other functionality for the user includes two lists of submissions: one list is for the submissions that are pending approval and the other for submissions that have been approved. Wherever a submission is listed, it includes an option to view details about the submission. The details of submission are available to both the user who uploaded the submission and to managers. The details include: the user's email address, the date and time the submission was uploaded, the date and time the submission was approved, the uploaded filename, the collection to submit to, a preview of the file contents, and an option to download the file that the user uploaded. The option to view details allows a manager to make the decision as to whether the user is allowed to submit to the collection that they are attempting to submit to and whether the content of the submission meets the submission policies of the institution, or not.

User	Filename	Submitted to
View Details Approve user@example.com	test-data-2015-10-30-15-58-11.csv	University of Cape Town > Completed > Masters
View Details Approve user@example.com	test-data-2015-10-30-15-58-27.csv	University of Pretoria > Completed > Masters

Figure 4. Screen snapshot of submissions pending approval list

A manager is presented with a list of submissions pending approval, as depicted in Figure 4. Each entry in the list includes: the user's email address, the collection to submit to and the options to view details, approve or remove the submission. Should a manager choose to remove a submission, the email address of the user that uploaded the submission is formatted as a mailto link. The mailto link will open up a new email in the manager's default email client with the recipient's email address and subject line already filled in. This enables an out-of-band communication between the user and the manager, should the manager have a query or wish to notify the user of a removal of a submission.

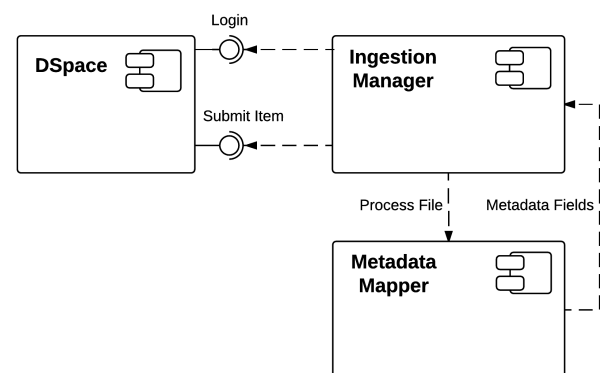


Figure 5. Component Diagram of connections between systems

3.2.4.1 Interface with Metadata Mapper

The Metadata Mapper (developed by Craig Feldman, and to be discussed further in his paper) is merged into the Ingestion Manager. Figure 5 depicts the connections between the Ingestion Manager, the Report Writer, and DSpace. The Metadata Mapper includes functionality to automatically determine the Dublin Core field type of each column of data in a CSV file. It also includes the option to correct any errors the automatic mapping process may have made. Login is handled by the Ingestion Manager, from there a user goes to a submission page that is shared by the Ingestion Manager and Metadata Mapper. The submission page, depicted in Figure 6, includes a form to upload a CSV file as well as for the user to enter other submission specific information. After the user submits the file, the Metadata Mapper processes the CSV file. The information from the Metadata Mapper is passed back to the Ingestion Manager, which then stores the submission and its associated information in the database.

Upload Batch

To get started, upload the file that contains the data you would like to ingest into a Collection. You can also save the metadata mapping for future use, or use a previously saved mapping.

Input file:
 No file chosen

Has header:
☐ The first line of the input file contains headings.

Field separator:
 Select...

Save mapping as:

Use previous mapping:

Collection:
 Select...

Figure 6. Screen snapshot of submission page upload form

3.2.4.2 Submission Workflow

The Ingestion Manager implements a workflow-based ingestion management system, similar to that which DSpace provides, whereby a manager has to approve a submission before it is entered into the repository. Refer to Figure 7 for an illustration of the process of the workflow.

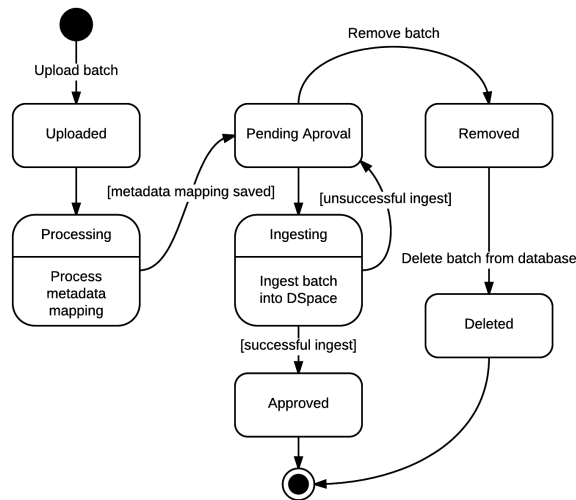


Figure 7. State Machine Diagram of submission workflow

A user uploads a batch of items to be ingested into the DSpace repository. The batch is a CSV file in which one line contains the metadata for a single item. The CSV file is typically an exported file from a legacy database. The submission is first passed to the Metadata Mapper to determine and set the Dublin Core fields that correlate to the columns in the CSV file. The information determined by the Metadata Mapper and collected from user input is passed to the Ingestion Manager. This information is then stored in the application's database. Thereafter, the submission is marked as pending approval and is then awaiting a manager to approve the submission.

While the submission is awaiting approval, both the user and the manager have an option to view details about or remove the submission. If a submission is removed, the submission details are deleted from the database. Alternatively, a manager has the option

to approve the submission, which will then ingest the batch into the DSpace repository.

3.2.4.3 Interfacing with DSpace

The Ingestion Manager interfaces with DSpace through the REST API made available in DSpace.

A collection ID is a unique ID used to identify a collection in DSpace. The collection ID of a collection in DSpace is needed in order to submit an item to a specific collection in DSpace. A call is made to the REST API to return an XML formatted list of communities. This list contains all the communities with their parent-communities (if applicable) and collections. From this a list of communities and collections is built, which helps the user identify the collection to submit to by name.

Ingesting items into DSpace is done through the REST API by submitting one item at a time. This is done because the current functionality of the REST API does not include a batch submission option. It also allows for reporting on specific items, should the ingestion fail. The process of submitting a batch involves iterating through the CSV file the user uploaded line by line and mapping the Dublin Core field types to the value in the associated column of the line. The REST API only accepts a JSON or XML formatted array of metadata items, where each metadata item is a text value and Dublin Core identifier pair. The Ingestion Manager uses JSON formatting when submitting to the REST API. Upon an unsuccessful submission, an error message, a preview of the failed entries, and an option to download a CSV file of the failed entries is displayed to the user.

[Download Report](#) [Try another report template](#) [Go to home page](#)

Report generate from template: Community Productivity

Report on The Productivity of Communities

This report details the productivity of communities based on their research outputs per type of output.

Number of Items In Each Community per Type per Month and Year

Count	Type	Community	Month	Year
35	Thesis	Current	October	2015
71	Thesis	Completed	October	2015

Number of Items In Each Community per Type

Number of Items In Each Community per Type

Figure 8. Screen snapshot of report preview screen

3.2.5 Report Writer

The initial design of the Report Writer was to implement an open source business intelligence solution. The problem with this approach is that, although the open source solutions provided many extra features, they were unnecessarily complicated for the requirements of this project and they lacked the direct integration with DSpace that we sought. So the decision was made to develop a reporting tool that had only a small subset of the features of the open source solutions, but was enough to meet the requirements.

The Report Writer generates reports from predefined templates. The templates are XML files (explained in Subsection 3.2.5.1) containing features which are translated to HTML components. Once logged in, a manager can select to generate a report from a list of available templates. Once the report has been generated, the manager is presented with a full preview of the report. The report preview screen is depicted in Figure 8. From the preview screen,

the manager can choose to download the report or return to the previous page to select another template.

Reports are generated in HTML format and are downloadable as a ZIP archive. ZIP archives were chosen so that the images can be bundled together with the HTML file in a single downloadable file. The option of embedding the images as Base 64 encoded images was investigated, but later dropped, because the embedded images were not visible in other applications.

Table 1. XML report feature descriptions

Feature	Tag value	Attributes
Page title	Text	
Heading	Text	Style (h1, h2, etc.)
Paragraph	Text	
Table	SQL SELECT query	
Pie chart	SQL SELECT query	Chart title, width and height.
Bar chart	SQL SELECT query	Chart title, width, height, x-axis and y-axis label.

3.2.5.1 XML Template Specification

The Report Writer was designed with a similar approach to other reporting solutions. That is, the layout and content of the report is specified in an external XML file. Report features (headings, paragraphs, tables, etc.) are specified in XML tags. Some tags allow for attributes to be set for extra customization. Table 1 lists each feature available, what the XML tag should contain and its optional attributes supported by the Report Writer. For features that contain SQL SELECT queries as the tag value, the database is queried and the results are displayed in the format specified by the feature.

Table 2. Information stored in applications' databases

Application	Stored in database
Ingestion Manager	Submission information.
Report Writer	Report template information.
Both	Which users are managers.

3.2.6 Database Design

An SQLite database is used by both the Ingestion Manager and the Report Writer. When selecting a database to use in the applications, different options were considered but SQLite was chosen because the applications require only small datasets and are likely to be of low volume use. SQLite is also beneficial in that it does not require a server to be run separately from the main application (unlike other RDBMSs, such as MySQL or PostgreSQL) and can be bundled together with the application [17]. Table 2 lists what the applications store in their SQLite databases.

To assign manager permissions to any DSpace user, a Python script has been provided for convenience. The Python script simply adds a user to the application's database and sets their role to manager.

3.2.7 User Interface Design

The design of the user interface for both the Ingestion Manager and the Report Writer is meant to resemble the DSpace 5 JSP UI. This decision was made so that users of DSpace 5 JSP UI will feel a familiarity when using our applications.

The user interface for both applications was designed using a user-centred design approach. A paper prototype of both applications was created and presented to a group of four masters and doctoral students from the Digital Libraries and Information and Communications Technology for Development labs at the Department of Computer Science at the University of Cape Town. The paper prototype consisted of an A4 page per screen of the UI with the components of the interface drawn in with pencil. Background information on the project was provided to the students and they were given use case tasks to complete.

Some of the feedback from the session included changing the wording of navigation buttons to be more descriptive, combining components of the user interface into logical groups, using tabs to separate long lists, and adding the option to see additional information. A digital copy of the paper prototype and a document containing feedback from the session can be found in the online supplementary materials.

4. IMPLEMENTATION

4.1 Initial Feasibility Demonstration

For the initial feasibility demonstration of the Ingestion Manager, different methods of ingesting items into DSpace were investigated. DSpace provides a command line interface through an executable, a service-level API and a RESTful web services API. The DSpace developers mailing list was consulted when faced with the decision of choosing the best interface and methods of authentication in DSpace. The recommendation received from a developer of DSpace was to use the REST API, as it has the most control when authenticating a user in DSpace and to not use the command line interface directly. Therefore, we choose to use the REST API when interfacing with DSpace.

For the initial feasibility demonstration of the Report Writer, the open source business intelligence reporting solution, ReportServer⁶, was implemented. Although it provided many useful features, it was decided that ReportServer was too complicated for the requirements of this project. ReportServer also presented errors that could not be diagnosed when connecting to the DSpace database. For these reasons, it was decided that we would develop our own reporting solution to meet the requirements of this project.

4.2 Software Development Methodology

The methodology used in this project is based on the basic principles of an iterative development lifecycle but adapted to fit a team consisting of only two people and for a short time frame. The phases of the development process included: planning, generating a prototype, getting feedback, and implementing the feedback.

During the planning phase, the DSpace mailing lists were consulted and the user survey was conducted. Following the planning phase, prototypes were developed. The initial feasibility demonstration prototype for the Ingestion Manager became an evolutionary prototype onto which functionality was added in increments. The initial feasibility demonstration prototype of the Report Writer became a throw away prototype because that prototype was discarded and a new application developed.

4.2.1 Central Unified Process

This project adopted ideas from the Central Unified Process during development. These ideas included: an iterative development cycle, tackle the high risk items early, regular engagement with our

⁶ <https://reportserver.net/en/>

project supervisor, regular verification of the quality of our software, model the software using UML, and manage our requirements.

Regular verification of the quality of our software was done by making use of unit testing and by seeking user feedback during development. Our requirements were managed by addressing the core features first and, only once they were addressed, did we consider additional functionality. We were not able to engage directly with the project's client on a regular basis due to the physical distance between us. We did, however, have regular meetings with our project supervisor to discuss our progress.

4.2.2 Scrum Board

A simplified scrum board was created to track the software development of this project. The scrum board consisted of a large sheet of cardboard paper divided into 3 sections; "to do", "in process" and "done." As new tasks of development were identified, they were written onto sheets of a note pad and stuck in the "to do" section. During the task's development they were moved to the "in process" section and once completed they were moved to the "done" section. Images of the scrum board taken during development can be found in the online supplementary materials.

4.3 Development Frameworks

4.3.1 DSpace

The applications were developed to work with DSpace version 5.2.

4.3.2 Backend

The backend of the Web application was developed using Java version 8. The backend consists of both Java Classes and Servlets.

Two versions of Apache Tomcat server were used during this project. For development, version 8 was used, and during testing version 7 was used.

4.3.3 Frontend

The user interfaces of both applications were developed using JavaServer Pages (JSPs) with HTML 5. Custom style elements were specified in CSS 3. JSPs were used because we wanted our applications to match the look and feel of the DSpace 5 JSP UI, which is built using JSPs. Some JavaScript was used in the applications as well as to ensure that some elements were reset when reloading the page. As in the DSpace 5 JSP UI, our applications used Bootstrap⁷ version 3 for all the HTML components and the default Bootstrap theme for styling.

4.3.4 Database

The Xerial SQLite JDBC⁸ library was used in both applications for accessing and creating the SQLite databases. All methods that contained SQL statements and variable user input made use of prepared statements when connecting to the application's database. This was done to help mitigate the effects of either malicious or accidental SQL injections.

Python was used when creating the script to add new users to the applications' databases. The script interfaces directly with the SQLite database file that is included with the application. When setting up the applications, only managers need be added to the database; all other users are treated as non-managers by default.

This is done so that there is no need to add every user of DSpace to the applications' databases.

4.3.5 REST API

Descriptions of each use and the endpoint of the DSpace REST API that was used are listed in Table 3. The host and port connection information to the REST API is specified in an external "properties" file. This is done to allow for the host or port of the REST API to change without the need to change programming code.

Table 3. Uses of the DSpace REST API

Application	Use of REST API	REST API endpoint
Ingestion Manger	Build a list of all collections.	/communities
Ingestion Manger	Submit an item to a specific collection.	/collections/{collection ID}/items
Both	User login.	/login
Both	User logout.	/logout

4.3.6 Ingestion Manger

The Ingestion Manager makes use of the JSON Simple⁹ library when compiling the JSON metadata arrays to pass to the REST API. A JSON metadata array of Dublin Core identifiers and text values is passed to the REST API endpoint. One JSON metadata array corresponds to one item in DSpace.

4.3.7 Report Writer

The Report Writer connects directly to the DSpace database to get a level of detail not available through the REST API. Connection configuration information for the DSpace database is specified in an external "properties" file to allow changes to the connection host, port, database name, username or password without the need to change programming code.

Only the PostgreSQL implementation of DSpace is supported by the Report Writer. The decision to not support the Oracle implementation was made because PostgreSQL is open source and being so is likely to have a greater user base than Oracle. Oracle also has licensing restrictions that proved troublesome when investigating the possibility of adding support for an Oracle database. The Report Writer accesses the PostgreSQL database of DSpace using the PostgreSQL 9.4 JDBC.

The JFreeChart¹⁰ library is used when creating charts to graphically display information.

4.4 Development Environment

4.4.1 Integrated Development Environment

The IntelliJ IDEA¹¹ Java integrated development environment (IDE) was used when developing the applications. The DSpace source code was setup as a Maven project in the IDE. Following the example of the DSpace source code, both applications are also Maven projects.

4.4.2 Version Source Control

BitBucket¹² was used for source code management throughout the project.

⁷ <http://getbootstrap.com/>

⁸ <https://github.com/xerial/sqlite-jdbc>

⁹ <https://code.google.com/p/json-simple/>

¹⁰ <http://www.jfree.org/jfreechart/>

¹¹ <https://www.jetbrains.com/idea/>

¹² <https://bitbucket.org/>

4.4.3 Project Management

The OpenProject¹³ project management solution at the Department of Computer Science at the University of Cape Town was used for project management and milestone deliverable submissions.

5. TESTING AND EVALUATION

5.1 Unit Testing

Apart from the informal testing completed throughout development, unit testing of selected methods was used to verify quality and consistency of the applications. The unit tests were written using the JUnit¹⁴ testing library. The focus of the unit testing was to test connections to other application's interfaces and whether the use of external libraries produced the results that were expected. Descriptions of the unit tests written can be found in Table 4.

Table 4. Description of unit tests for applications

Application	Unit Test
Ingestion Manager	Test the connection to the DSpace REST API used when building the collections list.
Report Writer	Test report generation, chart creation, and the connection to the DSpace database.
Both	Test the connection to the DSpace REST API for login and logout and to test the connection to the application's own SQLite database.

The unit tests proved particularly useful when merging the Ingestion Manager and Metadata Mapper components together, as they helped identify points where the two components needed to share information.

5.2 User Evaluation

A user evaluation was conducted after the first iteration of development to gather feedback from users. It involved presenting the development version of the applications to five Computer Science Honours students at the University of Cape Town and asking them to perform a set of tasks designed to test the core functionality of the applications. Most of the feedback the users provided concerned the user interface, however some feedback helped us improve the functionality the applications. The use case tasks and feedback from the user evaluation can be found in the online supplementary materials.

5.3 Performance Testing

Performance testing was conducted on both applications to determine whether the performance of the application degraded as the amount of work increased. Both applications were timed using the Java standard library's system time. The core piece of functionality for each application was identified and the difference between when that functionality ended, and when it began was used to make observations about the application's performance.

All performance testing was conducted on a computer with a 2,4 GHz Intel Core i7 processor and 8 GB 1600 MHz DDR3 RAM. Both the DSpace server and the PostgreSQL database server ran on the same computer that was used to do the performance testing.

5.3.1 Ingestion Manager

The core functionality of the Ingestion Manager is the process of ingesting a CSV file into a DSpace repository. This process includes: iterating through the CSV file line-by-line and mapping

the Dublin Core fields to the values in the columns of the line, generating a JSON array of the line, and passing that JSON array to the REST API endpoint.

An error in the DSpace source code was identified during performance testing (discussed in more detail in Section 6.2.1) which meant that the method of testing the Ingestion Manager had to be adjusted accordingly. Approximately 1 out of 100 lines would fail when attempting to ingest a batch into DSpace. To account for this error, the time required to ingest each line was recorded and if the ingestion failed the time spent on that line would be added to a cumulative total. This cumulative total was called the "error time" and is the amount of time a particular batch spent handling errors. The number of errors that occurred per batch was also recorded.

The performance testing of the Ingestion Manager involved ingesting large batches of items. Batch sizes ranged from 500 to 3000 lines and increased in steps of 500, but the metadata fields remained constant. Each batch was ingested five times and the amount of time taken to ingest the batch was recorded. In an attempt to reduce errors, the DSpace Web server and database server were restarted after each test trial. This test did not take into account the amount of time spent caching because the servers were restarted after each test trial so every trial would include time spent caching.

5.3.2 Report Writer

The core functionality of the Report Writer is the generation of a report, which includes parsing the XML template file and building the HTML report. The performance testing of the Report Writer involved recording the amount of time that it takes to generate a report depending on the number of items in the DSpace repository. The report template remained constant while the size of the repository was increased. The test included generating a report six times for each repository size. The sizes of the repository were 10, 100, 1,000, 10,000 and 100,000 items. The report was generated six times but the first trial was discarded to account for caching. An average of the remaining five trials for each repository size was calculated.

5.4 Usability Testing

In the usability test and acceptance test, when discussing the Ingestion Manager, included are both the Ingestion Manager and Metadata Mapper. This is because both components were merged together at this stage.

Usability testing was conducted to evaluate the usability of the applications we had developed. The usability testing was conducted with twelve Computer Science Honours students at The University of Cape Town. The students were presented with usability test tasks and asked to complete a questionnaire related to the usability of each application. The usability test tasks we created were based on the template available online from a book by Carolyn Snyder [16]. That template was chosen because of its similarity to a use case narrative. We used the System Usability Scale (SUS) designed by John Brooke as our usability questionnaire [5]. SUS gives a subjective assessment of the application's usability. It covers aspects of the need for training or support and the complexity of the system. It was used because it is a well established and tested methodology for testing the usability of a system. The results of SUS are used to determine a usability score.

5.5 User Acceptance Testing

The user acceptance test document we created was based on a template available from The Ohio Department of Higher Education

¹³ <https://www.openproject.org/>

¹⁴ <http://junit.org/>

[1]. This template was chosen because it was simple to adapt and easy for a user to understand. In our user acceptance test document, we listed the four functional requirements that we initially proposed to our client. With each functional requirement was the option to accept or reject it based on whether we had met the requirement. The document also included sections for the client to indicate whether the project met the initial objectives agreed upon and if it required any changes before it could be put into a production environment.

A server was setup to host the applications we developed so that our client could evaluate them remotely. For the evaluation, we setup an instance of DSpace on the server and used a PostgreSQL database already available to us. The source code repositories for DSpace, the Ingestion Manager and the Report Writer were cloned onto the server, configured and, using Maven, compiled to run on the server. Once the applications were available online we sent a SUS questionnaire, the user acceptance test document and instructions to our client.

6. RESULTS AND DISCUSSION

The full results of the usability testing and performance testing can be found in the online supplementary materials.

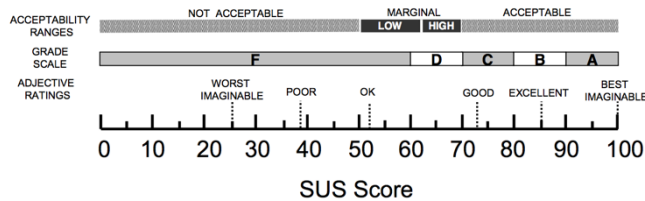


Figure 9. Comparisons between SUS, the adjective scale, the school grading scale, and the acceptability ranges scale [4]

6.1 Usability Testing

The results of the System Usability Scale (SUS) were recorded and analysed to produce a usability measure for the user interfaces of the Web applications. Scoring SUS produces a single value. The problem faced when using a single value to represent the usability of a system is how to interpret that single value. In a study conducted by Bangor, Kortum and Miller, where the relationship between the results of SUS, a seven-point adjective Likert scale and the school letter grading scale were compared to give a better understanding on how to interpret the results of SUS [4]. They had also previously proposed a set of acceptability ranges that help indicate whether a user interface can be considered acceptable or not [3]. Figure 9 is an image created by them to show the comparisons between the four scales. This will be used when discussing the results of the usability testing conducted.

6.1.1 Ingestion Manager

The Ingestion Manager achieved a SUS score of 84.167, which rounding to the nearest whole number is 84. Matching a SUS score of 84 to Figure 9, we can see that the Ingestion Manager scores just below “excellent” in terms of the adjective rating scale, a grade scale score of B and is well within the acceptability range’s scale of acceptable. Overall, the Ingestion Manager scored a very good result in terms of usability.

6.1.2 Report Writer

The Report Writer achieved a SUS score of 89.583, which rounding to the nearest whole number is 90. Matching a SUS score of 90 to Figure 9, we can see that the Report Writer scores within the “excellent” range in terms of the adjective rating scale, a grade scale score of A, and is well within the acceptability range’s scale

of acceptable. Overall the Report Writer scored an excellent result in terms of usability.

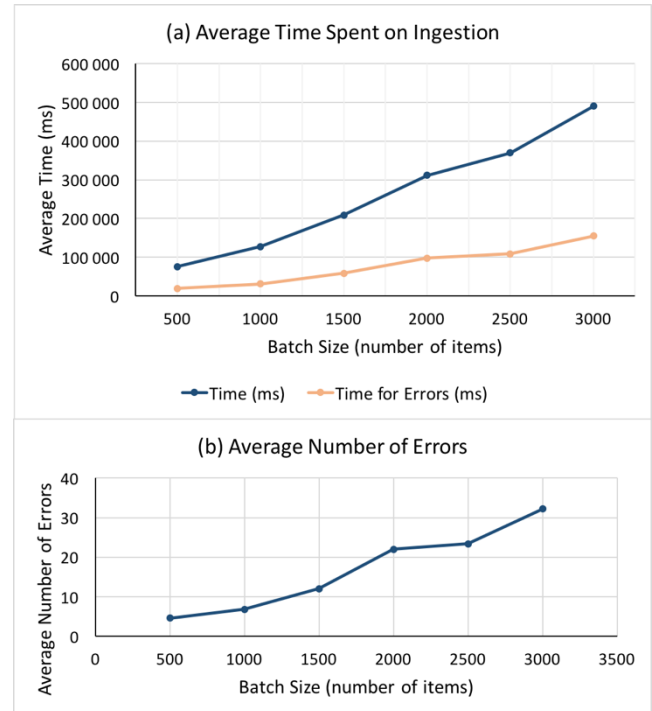


Figure 10. Ingestion Manager performance test results

6.2 Performance Testing

6.2.1 Ingestion Manager

The aforementioned error encountered whilst testing the performance of the Ingestion Manager had to do with the way in which DSpace handles database connections. After approximately 100 connections to the DSpace database the following connection would fail. This may be due to the fact that the database connections within DSpace are not cleared but the error was out of the scope of the Ingestion Manager development.

The results of the performance test for the Ingestion Manager show an approximately linear increase in the average amount of time it took to ingest a batch into DSpace. In Figure 10 (a), the darker line plots the average total amount of time the batch took to ingest while the lighter line plots the average total amount of time that was spent handling errors. In Figure 10 (b), the average number of errors that occurred during each ingestion also shows an approximately linear increase.

The results of the performance test of the Ingestion Manager show that, on average, there is no degradation in performance as the size of the batch increases. However, it did take a substantial amount of time to ingest large batches of over 1000 items. To help with the database connection error, the Ingestion Manager will retry once for a failed ingest. After the attempt to retry, if the item fails again, it will be added to an array of failed items, which is later made available to the manager to download as a CSV file.

6.2.2 Report Writer

The results of the performance test for the Report Writer are presented in Figure 11. The scale for both axes in Figure 11 are logarithmic and the graph depicts an exponential increase in the average time taken to generate a report on all items in the repository as the size of the repository increases. The report used in this test

was called “Community Productivity.” It was designed to showcase the productivity of each community in a DSpace repository.

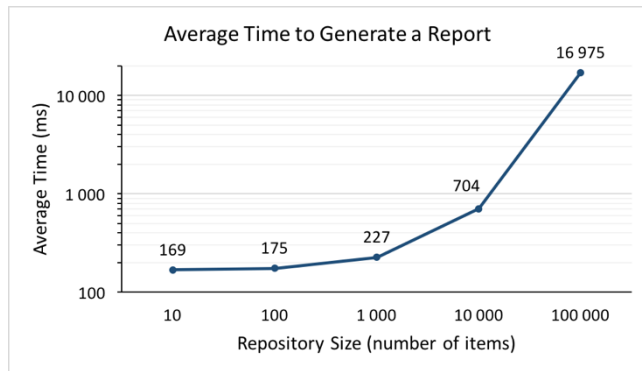


Figure 11. Report Writer performance test results

The results of the performance test of the Report Writer show that as the repository size increases, the amount of time needed to generate a report increases exponentially. To understand why the performance degraded so drastically a further test was done, but this time the report being generated was changed. The report used in this test was called “Collection Statistics.” It was designed to report basic statistics on the number of items in each collection. This report showed a large improvement in performance when compared to the previous test. For a repository size of 100,000 items, the report took on average 1,547 milliseconds to generate.

From these two tests we now know that the performance of the Report Writer is largely dependant on the complexity of the report being generated. The first report, “Community Productivity,” contained multiple complex SQL queries, each with multiple inner joins and aggregate functions. The second report also contained multiple SQL queries with inner joins but they were much simpler. These results show that the Report Writer can perform well when generating complex reports with smaller repository sizes of up to 10,000 items. However, for repositories greater than 10,000 item in size, there is a trade off in the time it takes to generate a report and the complexity of a report. To support complex SQL queries with large repository sizes, indexing could be used in the database. The performance of the query can be improved by indexing the fields used when joining tables.

6.3 User Acceptance Testing

Our client accepted that we had met all four of the functional requirements we had set out to meet. His comments on the Ingestion Manager were that it is “...faster and appropriate for content coming from HEIs [Higher Education Institutes].” He commented on whether the software produced met the objectives by saying that “The project is good, and it will expose research titles for both Masters and PhDs.” On the topic of whether there are any changes required before the software could be put into a production environment, he commented that “There [are] some fields that need to [be] enhanced before this can be in a production environment, i.e. date display (publication date) in the simple record.” Overall, the client approved the software we had created.

Both applications were given a score of 90 on the SUS questionnaires sent to our client. Matching a score of 90 to Figure 9 we can interpret that both applications score within the excellent range in terms of the adjective rating scale, a grade scale score of A and is well within the acceptability range’s scale of acceptable. The completed acceptance test document can be found in the online supplementary materials.

6.4 Difficulties Encountered

Of the difficulties encountered during this project, the three most notable are: the lack of detailed documentation for the DSpace API, the lack of documentation for DSpace’s database schema and the database connection error experienced during testing. When investigating DSpace API options, detailed documentation could not be found. This made the selection of API and development of the applications more difficult. DSpace’s database schema is not intuitive to understand nor is there documentation available to explain the rationale for the schema design.

7. CONCLUSIONS

Add-ons that help to transform DSpace from an institutional repository into more of a research information management system were developed. The add-ons were: a workflow-based ingestion management system called the Ingestion Manager and a report generation tool called the Report Writer. The Ingestion Manager allows for batches of items to be remotely submitted into a DSpace repository by non-administrator users. The batches are entered into a workflow-based system whereby managers are required to approve the submission before it is ingested into DSpace. The Report Writer was designed to enable detailed reporting on the objects in a DSpace repository. It borrows from the design of other business intelligence reporting solutions where XML report templates are used when generating reports.

An iterative software development methodology and user-centred design approach were used when developing the add-ons. A survey was conducted and the DSpace mailing lists were consulted to help guide the functional requirements of the add-ons. In addition, a paper prototyping session was held and user evaluations were done to help guide the user interface design of the applications.

The results from performance tests indicated that the add-ons have acceptable performance for increasing task size but perform best when used with smaller sized repositories. The results of the usability testing showed that we had designed and built software that is of a highly acceptable standard and easy for users to interact with. Feedback from the user acceptance testing indicated that we had met the functional requirements that we had set out to address.

We have achieved what we set out to accomplish, and that is to help transform an institutional repository, DSpace, to become more like a research information management system by identifying and addressing features that DSpace lacks.

8. FUTURE WORK

Possible future work on the Ingestion Manager would be: to implement an email notification system that alerts users on successful or failed submissions and alerts managers when there is a new batch pending approval, or to implement a queuing system to better handle large batches that completes the submission as a background task and alerts the manager when it is complete.

Possible future work on the Report Writer would be to perform security penetration testing. This is recommended as it connects directly to the DSpace database. If the Report Writer is vulnerable to SQL injection, for example, it could mean a compromise of the DSpace database is possible.

9. ACKNOWLEDGMENTS

I would like to thank our project supervisor, Assoc. Prof. Hussein Suleman, for his valuable input and guidance throughout our project. I would also like to thank our client, Lazarus Matizirofa from the NRF, for his input and for the test data he provided. Finally, I would like to thank my project team member, Craig Feldman, for help and cooperation throughout the project.

10. REFERENCES

- [1] Acceptance Test Template, 2003.
http://regents.ohio.gov/obrpmcop/forms/templates/temp_acceptancetesting.doc.
- [2] Adewumi, A. and Omoregbe, N. Institutional repositories: features, architecture, design and implementation technologies. *Journal of Computing*, 2, 8 (2011).
- [3] Bangor, A., Kortum, P. and Miller, J. The system usability scale (SUS): An empirical evaluation. *International Journal of Human-Computer Interaction*, 24, 6 (2008), 574-594.
- [4] Bangor, A., Kortum, P. and Miller, J. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4, 3 (2009), 114-123.
- [5] Brooke, J. SUS: A “quick and dirty” usability scale. In Jordan, P. W., Homas, B. T., Weerdmeester, B.A., Clellan, I. L. M. Ed. *Usability Evaluation in Industry*, Taylor & Francis, London, 1996, 189-194.
- [6] Garfinkel, S. MIT's DSpace Explained-Electronic repositories stretch to meet scholars' needs. *Technology Review-Palm Coast*, 108, 7 (July 2005), 50-51.
- [7] Heery, R. and Anderson, S. 2005. *Digital repositories review*. Joint Information Systems Committee JISC Review, London, UK.
- [8] InfoEd Global. What We Do – eRA 101 – InfoEd, 2012.
<http://infoedglobal.com/about-research-administration/what-we-do-%E2%80%93-era-101/>.
- [9] Ivanovic, L., Ivanovic, D., Surla, D. and Konjovic, Z. User interface of web application for searching PhD dissertations of the University of Novi Sad. In *IEEE 11th International Symposium on Intelligent Systems and Informatics*, (Subotica, Serbia, 2013), IEEE, 117-122.
- [10] Joint, N. Current research information systems, open access repositories and libraries: ANTAEUS. *Library Review*, 57, 8 (2008), 570-575.
- [11] Khan, S., Usman, A., Irfan, R. and Hayat, A. DSpace@SEECs: SEECs institutional repository system. In *Proceedings of 7th International Conference on Emerging Technologies*, (Islamabad, Pakistan, 2011), IEEE, 1-6.
- [12] Palmer, D. T., Bollini, A., Mornati, S. and Mennielli, M. DSpace-CRIS@ HKU: Achieving visibility with a CERIF compliant open source system. *Procedia Computer Science*, 33 (2014), 118-123.
- [13] REST API - DSpace 4.x Documentation - DuraSpace Wiki, 2013.
<https://wiki.duraspace.org/display/DSDOC4x/REST+API>.
- [14] REST API - DSpace 5.x Documentation - DuraSpace Wiki, 2013.
<https://wiki.duraspace.org/display/DSDOC5x/REST+API>.
- [15] Smith, M., Barton, M., Bass, M., Branschofsky, M., McClellan, G., Stuve, D., Tansley, R. and Walker, J. H. DSpace: An open source dynamic digital repository. *D-Lib Magazine*, 9, 1 (2003).
- [16] Snyder, C. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann Publishers, Boston, 2003.
- [17] SQLite. Appropriate Uses for SQLite, 2015.
<https://www.sqlite.org/whentouse.html>.
- [18] Tansley, R., Bass, M. and Smith, M. DSpace as an open archival information system: Current status and future directions. In Koch, T., Sølvsberg, I. Ed. *Research and advanced technology for digital libraries*. Springer, Heidelberg, 2003, 446-460.
- [19] Walsh, M. P. Batch Loading Collections into DSpace: Using Perl Scripts for Automation and Quality Control. *Information Technology & Libraries*, 29, 3 (Sept. 2010), 117-127.
- [20] White, C. The Next Generation of Business Intelligence: Operation BI. *DM Review*, 15, 5 (May 2005), 34-37.
- [21] Yeates, R. Institutional repositories. *Vine*, 33, 2 (2003), 96-101.