



**Building the
Digital Repository
of Ireland
Infrastructure**

dri[®]

Contributors:

Dr. Kathryn Cassidy, Software Engineer, Trinity College Dublin

Dr. Sandra Collins, Director Digital Repository of Ireland, Royal Irish Academy

Fabrizio Valerio Covone, Software Engineer, Dublin Institute of Technology

Dermot Frost, Manager, Trinity College Dublin

Damien Gallagher, Senior Software Engineer, Maynooth University

Dr. Stuart Kenny, Software Engineer, Trinity College Dublin

Eoin Kilfeather, Research Team Leader, Dublin Institute of Technology

Dr. Agustina Martínez García, Postdoctoral Researcher, Maynooth University

Charlene McGoohan, Requirements Manager, Maynooth University

Jenny O'Neill, DRI Data Curator, Trinity College Dublin

Sinéad Redmond, Software Engineer, Maynooth University

Jimmy Tang, Senior Software Engineer, Trinity College Dublin

Peter Tiernan, Systems and Storage Engineer, Trinity College Dublin

Dr. Sharon Webb, Knowledge Transfer Manager, Royal Irish Academy (former Requirements Analyst)

Edited by:

Rebecca Grant, Digital Archivist, Royal Irish Academy

Jenny O'Neill, DRI Data Curator, Trinity College Dublin

Dr. Sharon Webb, Knowledge Transfer Manager, Royal Irish Academy (former Requirements Analyst)

DOI: <http://dx.doi.org/10.3318/DRI.2015.5>

Contents


Director's Foreword	— 3
The Digital Repository of Ireland	— 5
Executive summary	— 6
1. Introduction	— 7
2. Requirements	— 13
3. Architecture of the System	— 17
4. Core Technology Choices	— 22
5. Storage	— 26
6. Metadata and Data Modelling	— 30
7. User Interface Design	— 47
8. Conclusion	— 56
Bibliography	— 57
Appendix 1	— 61
Appendix 2	— 62

Director's Foreword

The Digital Repository of Ireland is a national digital preservation repository for humanities and social sciences data. We started building the DRI e-infrastructure in 2011 at the inception of our project, and from the very beginning it has been a key priority for DRI to inform and be informed by international best practice. When we conducted research into what other national or large-scale digital repositories had chosen for their architecture, technologies and methodologies, we found that many organisations did not document and publish their technology choices. It is our ethos in DRI to openly share best practice with the wide community in the hope that we may all benefit from making informed choices based on commonly adopted standards and exemplars of good practice. Hence this report, which documents and explains the design methodology, the technology choices, the standards and reference models that we have implemented in the DRI.

A core principle of DRI is Open Access, and our choices of software and platforms have reflected this core principle; it is important to us, as a publicly funded endeavour, to implement community supported software, and to contribute to the community, and that is why we work with Hydra, Fedora, Solr and other open source code bases.

We don't present our infrastructure as the perfect solution or the only solution, but it is our working solution that we expect to demonstrate robustness, scalability and performance for our users, data depositors and partners. If you are designing a repository solution for your institution, your discipline or indeed your country, we recommend a detailed requirements analysis, and on the basis of that analysis make informed technology choices that meet your needs, whilst giving consideration to the learnings and implementations of the community and how you too might contribute into the global knowledge in this area.



I must thank our amazing infrastructure team in the DRI, past and present, who have worked tirelessly over the last three years to deliver the DRI repository. They have worked within a highly interdisciplinary team, and demonstrated great understanding and perseverance with the often complex and challenging requirements of a broad humanities and social sciences research and information management community.

We publish this report knowing that our work is not completed: a digital repository is a living, growing organism, that will face new challenges as the huge quantities of data grow even faster, as the complexity of the data increases, and as the requirements for sophisticated data visualisation, open data and research data management grow. This report is a snapshot in time of our development to date and I look forward to the next iterations that will reflect the enhanced technologies and approaches that will no doubt develop over the next years.

Dr. Sandra Collins
Director, Digital Repository of Ireland

The Digital Repository of Ireland

The Digital Repository of Ireland is an interactive, trusted digital repository for social and cultural content held by Irish institutions. By providing a central Internet access point and interactive multimedia tools, DRI facilitates engagement with contemporary and historical data, allowing the public, students and scholars to research Ireland's cultural heritage and social life. As a national digital infrastructure, DRI is working with a wide range of institutional stakeholders to link together and preserve Ireland's rich and varied humanities and social science data.

The DRI was established in 2011, when it received funding from the Irish Government's PRTLI cycle 5 for €5.2M over four years. The DRI consortium is comprised of the following partners: The Royal Irish Academy (Lead Institution), Maynooth University (MU), Trinity College Dublin (TCD), Dublin Institute of Technology (DIT), National University of Ireland Galway (NUIG), and National College of Art and Design (NCAD). The DRI is currently collaborating with a network of cultural, social, academic and industry partners, including the National Library of Ireland (NLI) and the Irish National Broadcaster RTÉ.

Executive summary

The Digital Repository of Ireland (DRI) is the national trusted digital repository for Ireland's social and cultural data. The repository links together and preserves both historical and contemporary data held by Irish institutions, providing a central internet access point and interactive multimedia tools. The successful implementation of DRI's fundamental goal, to build an interactive trusted digital repository (TDR), depends upon the system's ability to implement and satisfy identified user needs.

The architecture of the DRI infrastructure was designed to meet all of the requirements of the project and is broken into several different components, including a storage cluster, physical virtual machine (VM) servers and a virtualised environment to host the various services required to deliver the repository functionality.

In order to implement the DRI infrastructure a number of technology choices were made. A framework that occupies a middle-ground between a full repository solution and the basic components of a system was utilised. The Hydra framework was selected with Fedora being used for storing and managing ingested digital objects together with Solr for search and discovery and Blacklight providing the interfaces for user browsing, ingest and management functions.

In order to develop a Trusted Digital Repository it was necessary to build a robust and scalable storage solution. DRI decided to focus on software defined storage (SDS) solutions because of the cost, flexibility and sustainability they can provide, with Ceph being chosen for its selection of interfaces and APIs for many programming languages, as well as a rich array of libraries.

The underlying data management layer must be capable of dealing with a wide variety of digital collections. DRI encourages the use of best practice and commonly used data and metadata formats and standards which are stored in the Repository as collections of Fedora digital objects. To facilitate the creation, management and storage of digital objects¹, the Repository data models were implemented as a Ruby gem which makes use of ActiveFedora. ActiveFedora also facilitates the specification of relationships between digital objects.

As well as managing and preserving digital objects the Repository provides access to these objects. The Repository's user interfaces are realised using Ruby On Rails views and JavaScript framework JQuery. The complex adaptive layout of the Repository makes use of the SASS language and a grid framework called Zen Grids. Interactive multimedia tools include timeline and mapping visualisations.

¹ A digital object refers to a file or digital asset and its associated metadata e.g. an image, a document, an audio file and all its accompanying files, including xml encodings.

1. Introduction

A trusted digital repository (TDR) is “one whose mission is to provide reliable, long-term access to managed digital resources to its designated community, now and in the future” (Research Libraries Group and OCLC, 2002, p.5). The Digital Repository of Ireland (DRI) is committed to linking together and preserving both historical and contemporary data held by Irish institutions and providing a central internet access point and interactive multimedia tools for use by the public, students and scholars.

This report outlines the approach taken by DRI in addressing the technological challenges of building a TDR and it is hoped it will be of assistance to others who are developing digital repositories. It is aimed at a technical audience who are interested in the technical decisions made by the development team, but it may also be of interest to information professionals hoping to roll out similar programs in their organisation.

DRI was funded by the Higher Education Authority through its Programme for Research in Third Level Institutions (PRTL²), Cycle 5 (2011-2015). The project is federated across six academic institutions, Royal Irish Academy (RIA), Maynooth University (MU), Trinity College Dublin (TCD), Dublin Institute of Technology (DIT), National University of Ireland Galway (NUIG), and National College of Art and Design (NCAD). Since 2011 DRI has been awarded additional funding through Department of Arts, Heritage and the Gaeltacht, Enterprise Ireland, Irish Research Council, Science Foundation Ireland, among others. A report on long-term sustainability of digital repositories has also been published by DRI and discussions are on-going with various agencies to address DRI's sustainability (Kitchin, Collins, & Frost, 2015).

The consortium brings together researchers in computer science and software development, requirements engineering, UI design and experience, digital archiving, stakeholder engagement and community outreach, policy development and project management, and academics in the Humanities and Social Sciences (HSS). The project team work collaboratively to deliver a robust, digital platform that supports long-term digital preservation for deposited digital objects and provides sustainable access. The team also develops policies which directly inform the ongoing development and management of the Repository, which include but are not limited to areas such as copyright and licensing, data protection and digital preservation strategy (data citation, persistent identifiers, media formats and storage). The development of guidelines and training workshops, to engage with the DRI's designated community, is also an integral part of the team's work and an important mechanism to drive national best practice in the area of long-term digital preservation and data management.

² <http://www.hea.ie/en/funding/research-funding/programme-for-research-in-third-level-institutions>, last accessed 6 June 2015.

1.1. Project Structure

The work program of the DRI is organised into multiple Strands and Work Packages (WP).³ These are:

- Strand 1 Management – Long Term Planning (WP1) and Project Management (WP2)
- Strand 2 Context – Requirements Analysis (WP3) and Policies and Guidelines (WP4)
- Strand 3 Design and Implementation – System Architecture (WP5), User Interface Tools (WP6), Data Management (WP7) and Storage and Preservation (WP8)
- Strand 4 Rollout - Support, Access & Development – User Support, Training and Advocacy (WP9) and Demonstrator Projects (WP10)

While this report primarily focuses on the activities of Strand 3 and the Requirements Analysis Work Package (WP3), the development of the DRI infrastructure was carried out with input from all Strands and Work Packages as well as the various Task-forces and Working Groups within the project.

In parallel to the development work on the Repository, the DRI has produced a number of reports and guidelines on a range of topics including a review of digital archiving in Ireland (2012), an analysis of international approaches to caring for digital content (2013), guidelines for the use of metadata standards in the Repository (2015), fact sheets on metadata quality control and long-term digital preservation, and specific legal documentation.⁴ The DRI's reports, guidelines and fact sheets have influenced the development of the Repository and constitute an important output from the DRI's Strands and Work Packages.

The DRI's Demonstrator Projects (WP10) were designed to test various technical and policy components of the Repository as well as to showcase some of the HSS datasets which are managed by the consortium's partner institutions. Five demonstrator projects were included in the DRI project plan. Each project brought unique data types, copyright challenges, access and metadata requirements and provided necessary context for the development of the Repository.

The Clarke Studio Archive (TCD) consists of hundreds of drawings, notebooks and business records of the Harry Clarke stained glass studio. This collection consists of digitised text and images. Irish Lifetimes (MU) is a collection of oral and life histories that reflect the changing lives and times in Ireland from the foundation of the State. These life histories are captured in digital audio files together with transcriptions in text documents.

³ <http://www.dri.ie/about>, last accessed 6 June 2015.

⁴ <http://www.dri.ie/publications>, last accessed 6 June 2015.

Kilkenny Design Workshops Archive (NCAD) features a collection of digitised photographs, press cuttings and booklets recording the history of the Workshops and their impact on design in Ireland. The Irish Language and Cultural Heritage Project (NUIG) focuses on particular aspects of Irish cultural heritage, depicted through the medium of the Irish language. The NUIG collections contain digitised texts, images, audio and video which are drawn from a number of sources, including the University's own Irish language archives, as well as contributions by other external partners, including RTÉ Raidió na Gaeltachta. The Letters of 1916 (MU) is a crowd sourced collection of letters written around the time of the Irish 1916 Easter Rising. These have now been uploaded, transcribed and encoded by members of the public.

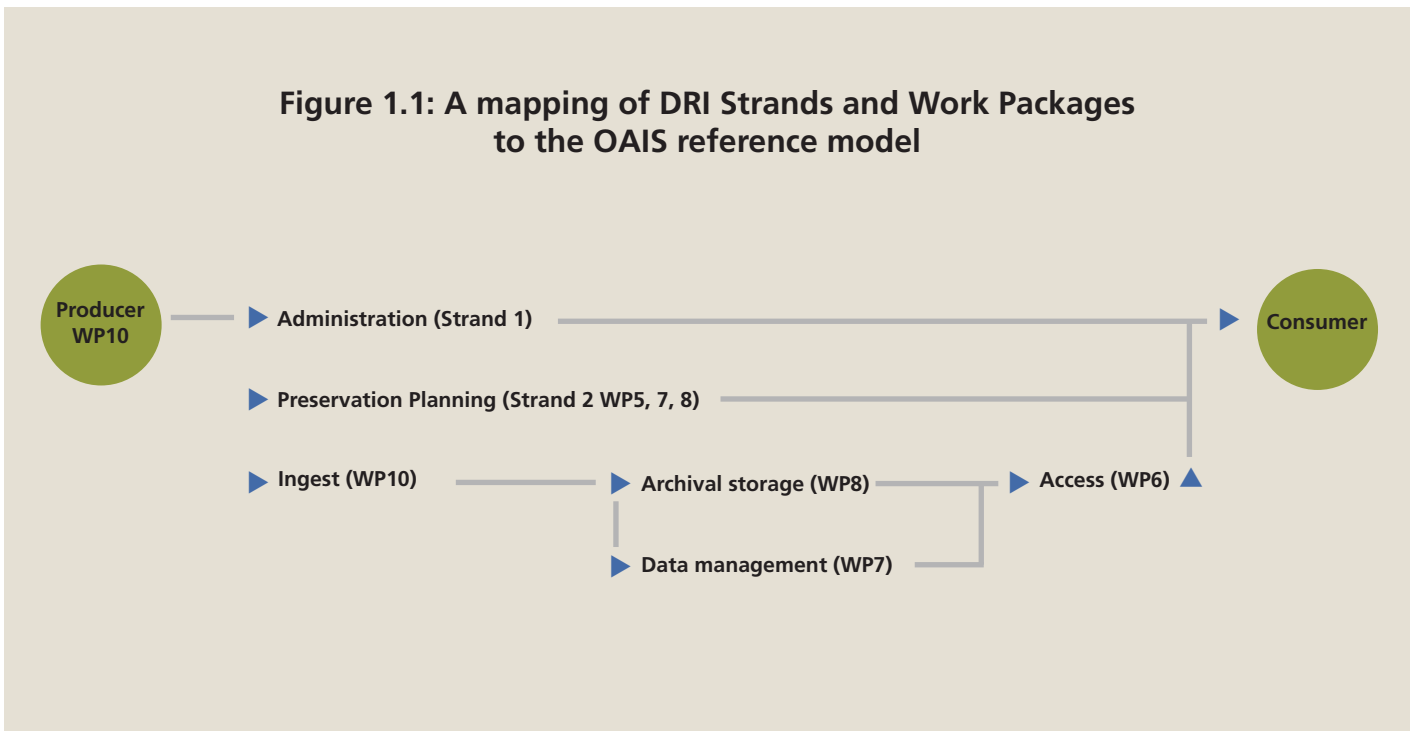
1.2. DRI and OAIS

The project structure of DRI was influenced by the Reference Model for an Open Archival Information System (OAIS). An OAIS consists “of an organization, which may be part of a larger organization, of people and systems that has accepted the responsibility to preserve information and make it available for a designated community” (Consultative Committee for Space Data Systems, 2012). The Reference Model establishes a common framework of terms and concepts which constitute a digital repository and lays out the functional components and responsibilities of such a repository at an organisational level.

The Strands and Work Packages of DRI can be mapped to the OAIS reference model as shown in Figure 1.1. The reference model recognises the interaction of producers of digital objects, management, the archive (or repository) itself and consumers of the digital objects. The archive consists of administration and preservation planning as well as functional entities: ingest, archival storage, data management and access.

Strand 1 are responsible for the long-term planning and project management for the Repository. Preservation planning is undertaken across the Strands, in particular by the System Architecture Work Package (WP5), the Data Management Work Package (WP7) and the Storage and Preservation Work Package (WP8). The Demonstrator Projects Work Package (WP10) are the producers of digital objects which are ingested into the Repository. The archival storage is the responsibility of the Storage and Preservation Work Package (WP8), while data management is the remit of the Data Management Work Package (WP7). User interface design and development to facilitate access is the responsibility of the User Interface Tools Work Package (WP6).

Figure 1.1: A mapping of DRI Strands and Work Packages to the OAIS reference model



1.3. Development Approach

As DRI was a new project the team did not have to be concerned with maintaining legacy systems or methodologies. The team adopted various agile techniques and methods and repeated those that appeared to work for DRI.

The agile methodology is an iterative approach to software development which means that requirements and solutions are not fixed at the start of the project, but rather evolve throughout the life of the project. Agile approaches emphasise collaborative planning, rapid prototyping and early delivery of features, continuous improvement, and encourages rapid and flexible response to change⁵.

DRI's work plan established that the Requirements Analysis and Policies and Guidelines Work Packages as well as the Design of Software Architecture Work Package began work at the same time. This created a potential conflict between the need of the Requirements Manager to define authentic user-focused requirements and the software development team's need to begin development of a technical solution in the absence of such requirements. However, as some of DRI's high-level business requirements were well defined from the outset of the project gave the software development team a starting point. Rapid prototyping was used to demonstrate features of the Repository

⁵ <http://agilemanifesto.org/>, last accessed 6 June 2015.

software. This allowed feedback from the Repository's stakeholders to inform both the requirements and development processes.

The use of an agile approach enabled the technical team to be responsive to changes in the project schedule. For example, supporting external curation tools developed by Inspiring Ireland⁶ required a re-prioritisation of features in order to develop the API functionality. Despite this change in focus, the disruption caused to the development effort was minimal given the agile methodology used.

At each new iteration or sprint of development a subset of requirements were prioritised by Strand 3 and Strand 2 at DRI's monthly cross-strand meetings. This forum enabled high-level feature specification to be carried out in collaboration with stakeholders and the demonstrator projects.

As a distributed development team, with responsibilities for different Work Packages shared among multiple consortium partners, a daily "stand-up" meeting by teleconference was essential. This provided a mechanism for the team to communicate, maintain an efficient working relationship and ensure issues, that could impact schedules, were identified and dealt with quickly.

Tools, including GIT, were used to assist the technical team in working collaboratively to produce code in an effective manner. GIT, a shared source code version control system was deployed to allow easy forking and merging of the code bases. Periodic code reviews were introduced to ensure quality and consistency of the code being developed across the consortium partners.

1.4. Report Structure

This infrastructure report details the various stages of development undertaken by DRI. It considers the work of the different Work Packages within Strand 3 as well as the Strand 2 Work Package, Requirements Analysis. Chapter 2, Requirements, details the requirements strategy developed by DRI and outlines the requirements gathering and analysis processes carried out by DRI.

The architecture of the DRI infrastructure was designed to meet all of the requirements of the project. This is outlined in Chapter 3, where both the system and application structures are described. This chapter includes a general overview of the DRI architecture as well as the architecture of the DRI application and the deployment architecture.

⁶ <http://www.inspiring-ireland.ie/>, last accessed 6 June 2015.

The core technology choices made to implement the Repository are then discussed. These choices were informed by the requirements gathering process and the architecture that DRI was aiming to implement. Key characteristics that influenced the decisions included the scalability and extensibility of the solutions examined, as well as ease of customisation, the system's ability to add data types and metadata standards and the range of storage options available.

The DRI approach to distributed and federated storage is addressed in Chapter 5. This includes a discussion of software defined storage (SDS) and solutions that were investigated by DRI. It also looks at the storage architecture, storage integrity and trust and migration of the entire storage solution should this become necessary if the system reaches the limits of its scale or the end of its supported life.

As part of the DRI's goal of building a TDR the underlying data management layer needed to be capable of dealing with a wide variety of digital collections. At the same time the data management layer was required to provide a set of data models that allow for the preserved data and metadata to be accessed and analysed by new tools developed in the future. Chapter 6 includes a discussion of the DRI data model and the metadata standards supported by it and the solutions developed by the Data Management Layer Work Package.

Chapter 7 outlines the design and implementation of the user interface. It also gives insight to the technologies used to implement the architecture of the UI for the Repository.

2. Requirements

The successful implementation of DRI's fundamental aim, to build an interactive trusted digital repository (TDR), is dependent upon the system's ability to implement its stated requirements and satisfy identified user needs. From the outset DRI committed to developing a system that reflects the authentic needs of its designated and diverse community of users.

The methodology for developing DRI's requirements was centred on a number of basic but important premises: – that the requirements are:

- driven by and focused on the user.
- considered within the humanities and social science problem domain.
- iterative and require input and feedback from the software development team, the Policy and Guidelines Work Package and end users (through demonstrator projects).
- flexible and dynamic without leading to software drift or requirements creep.

These assertions are reflective of DRI's mission to be user and community focused.

2.1. Requirements analysis and specification in DRI

In 2012 DRI released its first national report, 'Digital Archiving in Ireland: National Survey of the Humanities and Social Sciences', which published findings and observations on a number of issues including preservation, storage and formats, metadata and interoperability, as well as user tools and content management.⁷ This report was based on the requirements interviews DRI carried out to inform and develop DRI's requirements and policy statements. It was the first analysis of these interviews and provided the means to explore more thoroughly the issues, concerns and problems raised by DRI's community. The interviews, which informed the 2012 report, and their analysis, are reflective of DRI's requirements process for the period 2011-2013; the principal output of which was the formal specification of requirements which underpin, inform and prioritise the software development effort described in this report. DRI's formalised requirements statements are the accumulative effort of the elicitation, analysis and specification phases of the requirements process.

DRI's requirements range from high-level business requirements to functional requirements that detail user interactions with the repository and its content. The first and most important requirement, that the repository should be a TDR and should provide a range of research tools, came directly from the project's strategic objectives.

⁷ O'Carroll, A and Webb, S., 'Digital Archiving in Ireland: National Survey of the Humanities and Social Sciences' (2012) available <http://dri.ie/digital-archiving-in-ireland-2012.pdf>, last accessed 6 June 2015.

As well as being influenced by the OAIS reference model (see Section 1.2), DRI adopted the Data Seal of Approval (DSA, Data Seal of Approval Board, 2013) as a guideline for its policies, and also consulted ISO 16363⁸ standard for Trusted Digital Repositories, which is based on the Trustworthy Repositories Audit & Certification (TRAC, Online Computer Library Center & Center for Research Libraries, 2007) for additional guidance. These factors introduced a number of requirements specifically related to the provision of a TDR, such as the requirement to generate checksums of ingested files, to have backup and disaster recovery processes in place, and to generate audit reports.

Requirements were informed by the legislative framework in which the repository would operate. These included requirements related to the EU Cookie Directive (European Parliament, Council of the European Union, 2009), Copyright and Intellectual Property law (Copyright and Related Rights Act, 2000) and support for Freedom of Information requests (Freedom of Information Act, 1997).

As part of the requirements gathering process a review of emerging approaches to caring for digital data (O'Carroll, Collins, Gallagher, Tang & Webb, 2013) was performed to examine national and international best practices in the field. This informed not only requirements and policy, but also the technical and software decisions discussed in the following chapters of this report.

As stated, DRI is committed to satisfying authentic user needs. In order to achieve this it was necessary to determine the target community and identify the different actors and users of the system (e.g. depositor, collection manager, expert user, novice user, public user, third level researcher, external repositories, etc.). Different stakeholder personas were developed through stakeholder and user group analysis as well as from interviews, resulting in the development of use cases and use case scenarios. DRI's demonstrator projects also provided an excellent opportunity for the requirements, Policy and Guidelines and development Work Packages to engage with authentic users of the system. This process generated open discussion on user needs and revealed important user and functional requirements related to basic operations that users expected to be able to perform on the repository. Ingestion, editing, creating collections, publishing and others are examples of such functional requirements.

In addition to requirements that mapped to specific activities within the repository, there were non-functional requirements, such as the requirements for the repository to be robust, scalable and secure. The DRI storage and performance metrics survey helped to identify the typical storage and performance requirements of our stakeholders.

⁸ http://www.iso.org/iso/catalogue_detail.htm?csnumber=56510, last accessed 6 June 2015.

Performance testing of the system then allowed the project to measure success in meeting these requirements.

In order to be successful any software product must consider usability issues. These introduced further requirements such as the need to support common browsers, to be accessible, and to work on multiple devices (PC, smartphone, tablet, etc.).

Some requirements related specifically to DRI's intention to interact with Irish and EU projects, these included a REST Application Programming Interface (API) to support harvesting of digital objects from DRI.

2.2. Requirements management in DRI

The overall requirements engineering process is iterative and entails 'multiple cycles' between requirements elicitation, analysis, specification and, of course, validation (Wiegiers, 2006, p9). The use of a central requirements management system (RMS), CaseComplete⁹, supported this iterative model and ensures that the requirements were accessible across the Strands. The RMS also assisted transparency and traceability within the requirements process, an important feature given the nature and scope of DRI.

DRI's RMS was managed, updated and populated by DRI's Requirements Analyst and subsequently the Requirements Manager. This ensured continuity and was essential to the requirements management strategy. Contributions, feedback and comments on requirements, from the various team members in all Strands, were also an essential feature of the iterative requirements process. Regular requirements reviews and updates were given at cross-strand meetings and in the last year of development a bimonthly review was carried out with members of the development team to review the implementation status of the functional requirements. A change history was documented within each requirement statement in the system. No requirements were deleted - instead redundant requirements were tagged obsolete and remained in the system for continuity, demonstrating the evolution of the requirements and the Repository.

2.3. Requirements implementation

In order to ensure that the development work proceeded directly from the project requirements, each high-level requirement had to map to a set of features that could be implemented within the repository application.

⁹ <http://casecomplete.com/>, last accessed 6 June 2015.

The requirements were thus translated into executable requirements or specifications. Described as ‘outside-in development’ (Wynne & Hellesøy, 2012, p. 18) these executable specifications were composed of step-by-step descriptions of how tasks would be completed within the Repository. The executable specifications formed not only a detailed specification of the Repository’s functionality, but they also defined the acceptance tests for the system. Acceptance tests are expressions of ‘what the software needs to do in order for the stakeholder to find it [the system] acceptable’ (Wynne & Hellesøy, 2012, p. 4). This is part of the behaviour driven development (BDD, Chelimsky et al, 2010) approach to software development that focuses on cooperation between the stakeholder and the software development team. BDD is part of the agile software development methodology.

The Cucumber¹⁰ framework was used to translate the project requirements into the executable specifications and acceptance tests for the developers in DRI. An example of such a cucumber feature file is given in Appendix 1. The Cucumber feature files were then augmented with RSpec¹¹ ‘step definitions’ that translated the human-readable Cucumber feature specification into an executable test that could be run to validate the feature. The combination of a testing framework and continuous testing meant that the team was able to change code without fear of introducing conflicting or defective changes.

The Cucumber features are written and expected inputs and outputs are specified before the function or method being tested has been developed. This approach allowed the developers to view the methods and functions from the user’s point of view; from the vantage point of a user’s tasks and goals, rather than from the developers, solution-based, perspective. It also ensured that the technical team was delivering on end-user requirements.

¹⁰ <http://cukes.info/>, last accessed 6 June 2015.

¹¹ <http://rspec.info/>, last accessed 6 June 2015.

3. Architecture of the System

The architecture of the DRI infrastructure was designed to meet the requirements of the Repository as outlined in Chapter 2. As a Trusted Digital Repository (TDR) the infrastructure must ensure reliable, long-term, preservation of digital objects. However as a key aim of the Repository is also the accessibility of content, it is necessary to provide a comprehensive, fast and easy-to-use user interface with appropriate research tools. While frameworks adopted by the DRI, such as OAIS (see Section 1.1), provide guidance on the responsibilities and functions of a TDR, they do not specify any technical solutions, acting only as a high-level guide to some of the types of services that should be included.

3.1. General overview of the DRI architecture

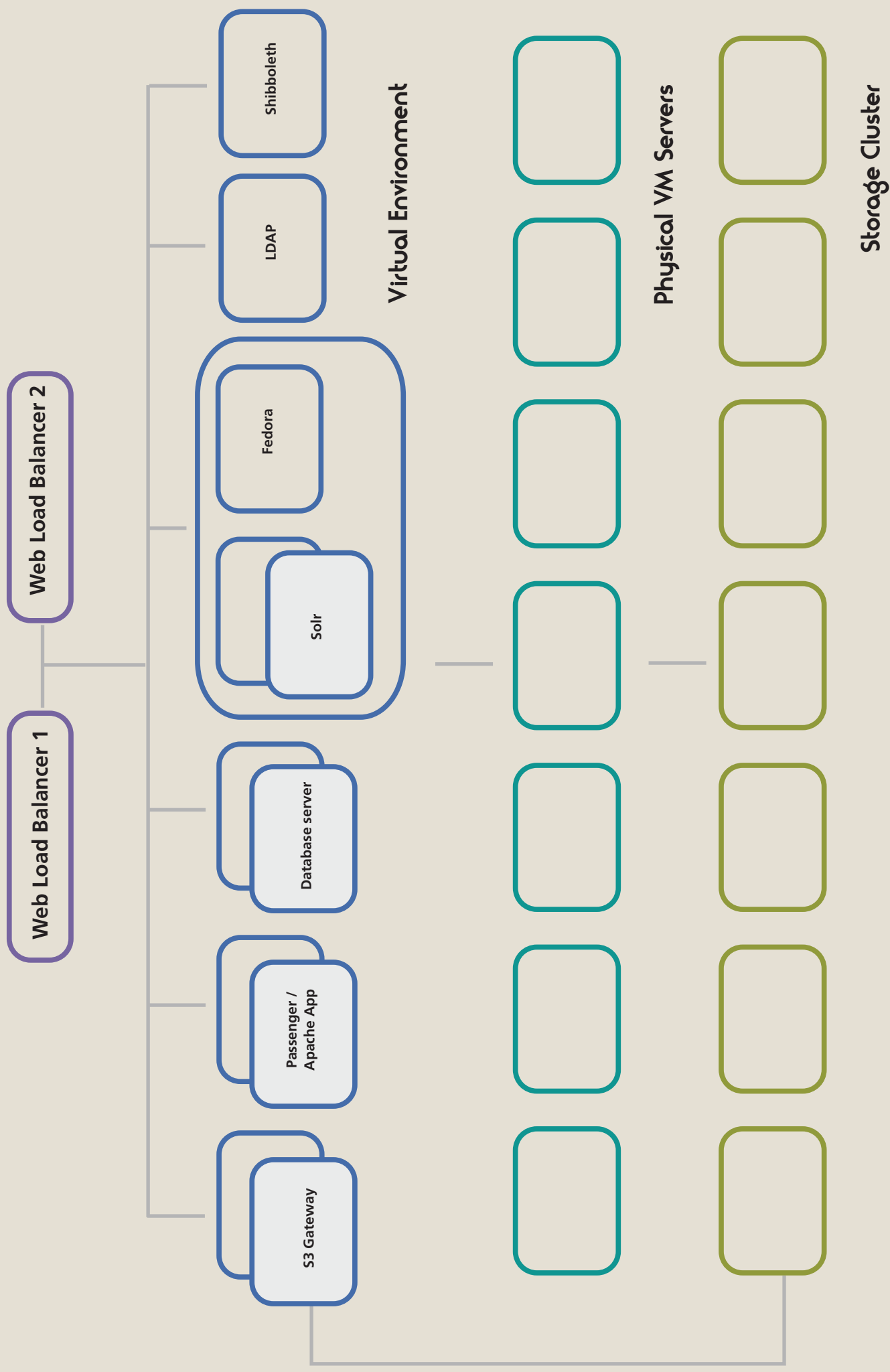
The DRI infrastructure is broken into several different components. A storage cluster (see Chapter 5) provides storage and preservation services for the digital objects, as well as storage for virtual machine images, documentation and other shared data. The storage is distributed across multiple geographically separate sites to provide failover and disaster recovery.

A number of physical servers act as virtual machine hosts running a virtualised environment. These are also located across multiple sites. Virtual servers running within the virtual environment host the various services required to deliver the Repository functionality. These services include application servers, databases, search and repository services, authentication services and storage access services. The majority of these services are replicated within the virtual environment for scalability and failover purposes.

Load balancers provide for scalability and high-availability both within a single site and across multiple sites. The entire infrastructure is replicated in a secondary site to allow for failover and disaster recovery of the whole DRI system.

The architecture of the DRI Infrastructure is shown in Figure 3.1.

Figure 3.1: Major components of the DRI Trusted Digital Repository



3.2. Architecture of the DRI application

The Model-View-Controller (MVC) pattern was chosen to implement the Repository. MVC separates the presentation of data in the system from the back-end representation and storage of the data. This separation allows for independent components to provide the required functionality of the system and suited the distribution of work among the project's Work Packages. It also makes it easier to replace components without disrupting the rest of the system and allows for the provision of multiple components with, for example, different views of the same data for different user types or use cases.

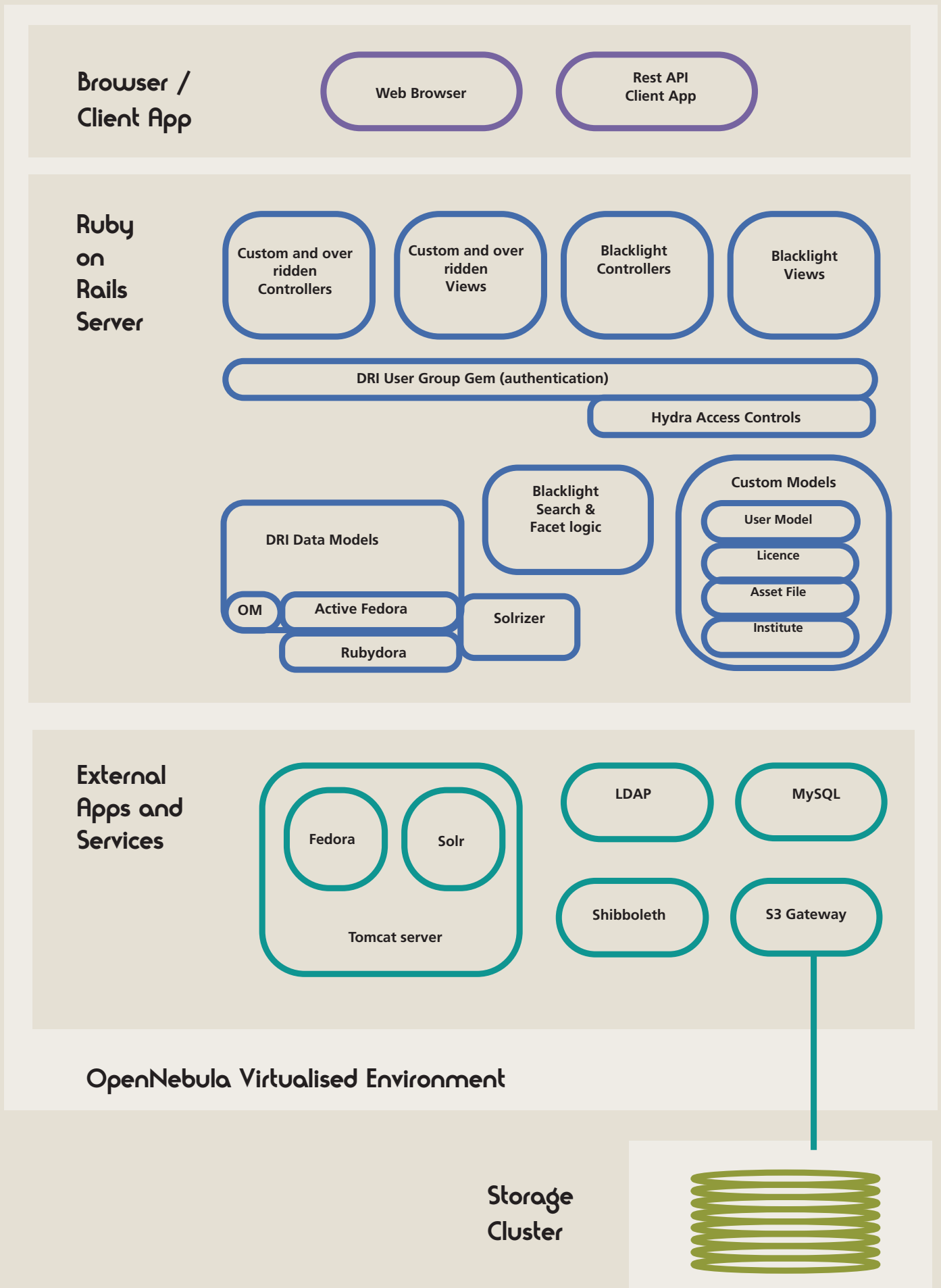
The Model components contain a description of the DRI content model which can be instantiated by the controllers, and which is specific to DRI's needs. The models provide a set of rules for the digital objects and collections outlining the required metadata fields and various mappings of formats and standards into the DRI content model. This modelling work is one of the primary tasks of the Data Management Layer Work Package (see Chapter 6). Other models describe users, institutes, licences and other entities within the system.

The View components in the DRI application are primarily the discovery and administrative interfaces. These are designed and implemented mainly in the User Interface Tools Work Package. The DRI repository also exposes an API for programmatic access. The API follows the Representational State Transfer (REST) design principle, where the HTTP methods, POST, GET, PUT, and DELETE, are explicitly mapped to the create, read, update, and delete (CRUD) operations of the Repository.

The Controller components contain the business logic which governs how actors interact with digital objects within the TDR. These interactions are defined by requirements and policy and are implemented between the User Interface Tools and Data Management Layer Work Packages as appropriate.

The application stack is illustrated in Figure 3.2 while a detailed class diagram for the application is given in Appendix 2.

Figure 3.2: Architecture of the DRI Trusted Digital Repository Application



3.3. Deployment architecture

In order to isolate the various services required to deliver a TDR, each service is deployed into its own virtual machine within the virtual environment. By isolating each service in this way operational or security issues that need to be addressed for a particular service, for example upgrading a service or applying a security patch, will not affect other TDR services.

A private cloud is used to deploy the DRI application together with the supporting components and databases. This system is based on OpenNebula¹², which provides a scalable, feature-rich virtualisation stack for Infrastructure as a Service (IaaS) installations (Mell & Grance, 2011). The storage infrastructure does not run within OpenNebula for performance, reliability and robustness reasons.

DRI aimed to ensure that the deployment process used to install and configure its hosts and software is robust and repeatable. A robust and repeatable deployment process allows the DRI systems engineers to deploy multiple replicas of any given service for backup or scalability purposes. It also ensures a rapid redeployment in the case of catastrophic failure. To this end the testing and deployment process of all system components is fully automated. The physical and virtual hosts are configured using Ansible¹³, which is an agentless configuration management, application deployment and automation tool.

DRI also uses a continuous build and test system to ensure that all new code changes that are committed to the software repository are automatically tested. In addition, based on configuration options, a new version of the codebase can be deployed automatically if these tests pass. DRI uses Buildbot¹⁴, an open source application that runs repeated jobs, such as building applications, to run tests and generate reports. This automation of building and testing components increases the team's productivity by providing regular feedback to the developers.

¹² <http://openebula.org/>, last accessed 6 June 2015.

¹³ <http://www.ansibleworks.com/>, last accessed 6 June 2015.

¹⁴ <http://www.buildbot.net/>, last accessed 6 June 2015.

4. Core Technology Choices

In order to implement the DRI infrastructure a number of technology choices were made. These choices were informed by the requirements gathering process and the architecture that the DRI was aiming to implement. Key characteristics based on the DRI's requirements that were pivotal in choosing a repository system included:

- Scalability and extensibility
- Ease of customisation
- Ability to add data types and metadata standards
- Range of storage options

Due to the large scope of the project, emphasis was put on scalability and how extensible each software or storage solution was. In terms of the Repository software two possible approaches were considered: the team could either enhance and customise an existing repository solution to meet the specific needs of the DRI, or build a new bespoke solution by combining existing search and repository technologies.

4.1. Repository solutions

A number of existing repository solutions were considered. These mostly aim to provide an 'out of the box' solution, with minimal install and configuration necessary. Examples include commercial systems, such as CONTENTdm¹⁵ originally developed by the University of Washington (Zick, 2009), DigiTool¹⁶ from Ex Libris, as well as open-source systems such as DSpace¹⁷ which is developed under the stewardship of DuraSpace and EPrints¹⁸ created by the University of Southampton.

These systems are capable of scaling to accommodate large collections as would be required by DRI. Most also offer some level of extensibility and customisation, however, this tends to be more focused on the User Interface (UI), rather than services and functions such as ingest and storage. Additionally, the long-term continued availability and support of commercial, closed source products is not guaranteed. While this is also true for open source systems, the availability of the source code means that it may be possible to continue development of such systems in-house.

¹⁵ <http://www.contentdm.org>, last accessed 6 June 2015.

¹⁶ <http://www.exlibrisgroup.com/category/DigiToolOverview>, last accessed 6 June 2015.

¹⁷ <http://www.dspace.org>, last accessed 6 June 2015.

¹⁸ <http://www.eprints.org/>, last accessed 6 June 2015.

4.2. Repository framework

Given the limitations described in Section 4.1, together with the complicated workflows required, the mix of user types expected and the number of metadata standards to be supported, building a new solution was the preferred option. Three basic components were required for the Repository: a system for storing and managing ingested digital objects; a search technology for discovery; and an application providing user browsing, ingest and management functions.

Although a custom solution was seen as the best way of meeting the requirements, building the system from basic components would have required significant development effort. Given the resources available an alternative, hybrid, approach was found, whereby a framework that occupies a middle-ground between a full repository solution and the basic components of a system was utilised. The framework chosen is called Hydra¹⁹.

The risk involved with depending on commercial software, particularly for long-term preservation activities, has already been mentioned. A similar risk could be said to exist with open-source software, such as Hydra, where long-term continued availability and support of the product is not guaranteed. This is somewhat negated by Hydra's strong community involvement, in the core development activities and in its adoption and usage. Many large institutions have committed to supporting the project as Hydra partners, including Yale University, Virginia Tech and Princeton University Library. The National Library of Ireland and University College Dublin are also working with Hydra and its components.

By actively participating in this community the DRI has gained from both the experience of the community and pre-existing state of the art. Re-using tools and libraries contributed by others, such as Sufia²⁰ and Hydra Derivatives²¹, allowed the development of the Repository to proceed at a much faster pace than would have been possible had the system been developed independently.

Project Hydra is a multi-institutional collaboration steered by Stanford University, University of Hull, University of Virginia, Duraspace and Data Curation Experts. The Hydra technical framework provides an 'ecosystem of components'²² on top of which it is possible to construct a full-featured repository application. Together these components allow for the Create, Read, Update, and Delete (CRUD) operations required by a repository.

¹⁹ <http://projecthydra.org/>, last accessed 6 June 2015.

²⁰ <https://github.com/projecthydra/sufia>, last accessed 6 June 2015.

²¹ <https://github.com/projecthydra-labs/hydra-derivatives>, last accessed 6 June 2015.

²² <http://projecthydra.org/>, last accessed 6 June 2015.

The primary platforms of the Hydra framework that support these operations are Fedora²³, Solr²⁴, Blacklight²⁵ together with the core Hydra libraries²⁶. These are described in Section 4.3.

4.3. Repository components

The Hydra framework provides a set of libraries (known as Ruby gems²⁷) that can be built on to implement a complete repository solution. Applications can be developed rapidly on top of the Hydra gems using the open-source web framework Ruby on Rails²⁸. Ruby on Rails is a very scalable development platform. It has been used to build large high-traffic websites with great success, for example, Twitter and Github²⁹. Together, the Gems produced by Project Hydra are intended to significantly reduce the need to spend development resources on constructing the framework of a repository, instead allowing developers to focus on developing data models and other services.

Applications built in this way are known as 'Hydra Heads', where Fedora together with Solr represent the 'body' of a Hydra repository, i.e. the content, while the 'head' is the application layer that connects users to the 'body'. It is possible to have several Hydra Heads connected to one body, for example one application for normal users (the consumers of digital objects) and a separate management application for depositors (the producers of digital objects).

Flexible Extensible Digital Object Repository Architecture (Fedora) is open source software designed as a core repository service for the storage, management and access of digital objects. It provides SOAP and REST application programming interfaces (APIs) for software development. It is designed to work with any software architecture and to integrate with a large variety of storage solutions, from common SQL databases to advanced low-level storage solutions like iRODs, Amazon S3 and SRB. By default it provides Dublin Core metadata for each digital object, but it allows developers to add any metadata format to a digital object. Fedora can integrate with other software services using its Content Model system for describing object types (Davis and Wilper, 2011).

Solr is a Java-based search engine from the Apache Software Foundation. It uses the Apache Lucene³⁰ search engine library in its core indexing and searching algorithms.

²³ <http://www.fedora-commons.org>, last accessed 6 June 2015.

²⁴ <http://lucene.apache.org/solr/>, last accessed 6 June 2015.

²⁵ <http://projectblacklight.org/>, last accessed 6 June 2015.

²⁶ <https://github.com/projecthydra/hydra-head>, last accessed 6 June 2015.

²⁷ <http://guides.rubygems.org/what-is-a-gem/>, last accessed 6 June 2015.

²⁸ <http://rubyonrails.org/>, last accessed 6 June 2015.

²⁹ <http://rubyonrails.org/>, last accessed 6 June 2015.

³⁰ <http://lucene.apache.org>, last accessed 6 June 2015.

Both Lucene and Solr share the same development team. Lucene provides Solr with a very advanced searching text analysing filter. This allows the software to be configured to perform complex searches involving boolean expressions, wildcards, date ranges, text ranges, 'sounds like' text matching and geospatial searching. Out of the box, Solr provides highlighted snippets, multi-select facet filtering, auto-suggest where it provides suggestions to the user on how to complete a query and spell checking to the user³¹. Solr can also automatically extract and index the internal text and metadata of common proprietary data formats such as PDF and Microsoft Office documents using Apache Tika³².

Hydra uses Blacklight, a discovery framework, to provide a standard user interface that allows for search and display of objects stored in the repository. Blacklight provides features such as faceted-search and browsing, but is also highly customisable. There are many examples of interfaces built using Blacklight, for example, The Rock and Roll Hall of Fame³³ and the Stanford University Library Catalog³⁴.

³¹ <http://lucene.apache.org/solr/features.html>, last accessed 6 June 2015.

³² <http://tika.apache.org/>, last accessed 6 June 2015.

³³ <http://catalog.rockhall.com/>, last accessed 6 June 2015.

³⁴ <http://searchworks.stanford.edu/>, last accessed 6 June 2015.

5. Storage

To achieve high levels of trust and preservation in the Repository it was necessary to build a storage infrastructure that could satisfy DRI's requirements. These included the need to have independent, replicated and fault tolerant storage pools, geo-replicated copies, tape cold storage and integrity checks. The foundation of this is making the correct technology choices. A full preservation strategy will be published separately to this report.

5.1. Storage solutions

In order to choose the most appropriate storage for the DRI, testing of storage solutions was performed and consisted of full hardware installation, adding/removing storage, high availability and feature evaluation. Requirements such as reliability, scalability, federation and interoperability were desired. Specific features such data distribution, replication, scrubbing/self-healing and complete APIs are useful in providing ongoing, highly-available access.

The DRI decided to focus on software defined storage (SDS) solutions because of the cost, flexibility and sustainability they can provide. Software defined storage is a form of storage virtualisation whereby the underlying physical hardware (servers and disks) is separated from the software that manages the storage. The SDS software provides access to the storage hardware through standardised APIs and interfaces such as POSIX shared filesystems, REST/S3 compatible gateways and/or network block devices.

Four SDS solutions were investigated: GPFS³⁵, Hadoop HDFS³⁶, iRODS³⁷ and Ceph³⁸. All of these run on commodity servers, removing vendor lock-in and thus the team was free to choose any hardware from any manufacturer. These SDS solutions are available for many Linux distributions and some run on a Windows server which increases flexibility. All are highly scalable and are established, well maintained projects that are supported by large corporations that use and contribute to them.

Of the four solutions evaluated Ceph was deemed to be the most suitable fit for DRI's needs. It offers the best and most complete selection of interfaces and APIs; REST/S3 compatible gateway, network block devices, POSIX filesystem as well as a rich array of libraries and APIs for many programming languages, as identified from the storage evaluation. It offers total data reliability with excellent data distribution, replication and self

³⁵ <http://www.ibm.com/systems/software/gpfs>, last accessed 6 June 2015.

³⁶ <http://hadoop.apache.org>, last accessed 6 June 2015.

³⁷ <http://irods.org>, last accessed 6 June 2015.

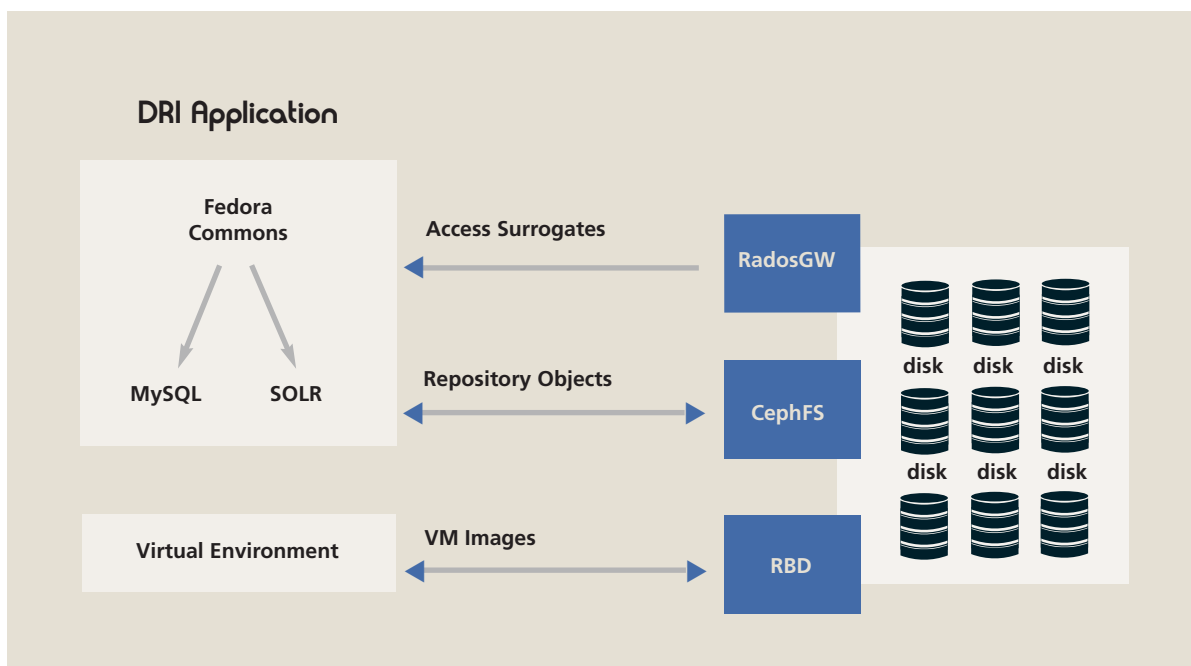
³⁸ <http://ceph.com>, last accessed 6 June 2015.

repair. It provides petabyte scalability with no single points of failure. It is complete; all components are developed and maintained by the Ceph project, no external services or software are required. Finally, it is free and open source, well maintained and is backed by the Red Hat corporation, who are committed to its long term success.

5.2. Storage architecture

Two technologies are used to manage storage, Ceph and Bareos³⁹. Ceph provides DRI with cloud-like, storage-as-a-service. It is used for hot or live data storage for virtual machine (VM) images and repository data. Bareos is used for cold tape storage and backup.

Figure 5.1: DRI storage architecture



At its lowest level, Ceph provides storage to the virtual environment, OpenNebula (see Section 3.3), as a place to store VM disk images over the Rados Block Device⁴⁰ (RBD) interface. The repository application uses the Ceph shared filesystem, CephFS⁴¹, to store archival objects while the discovery user interface uses the REST/S3 compatible Rados Gateway (RadosGW⁴²) to deliver surrogates of these archival objects to users.

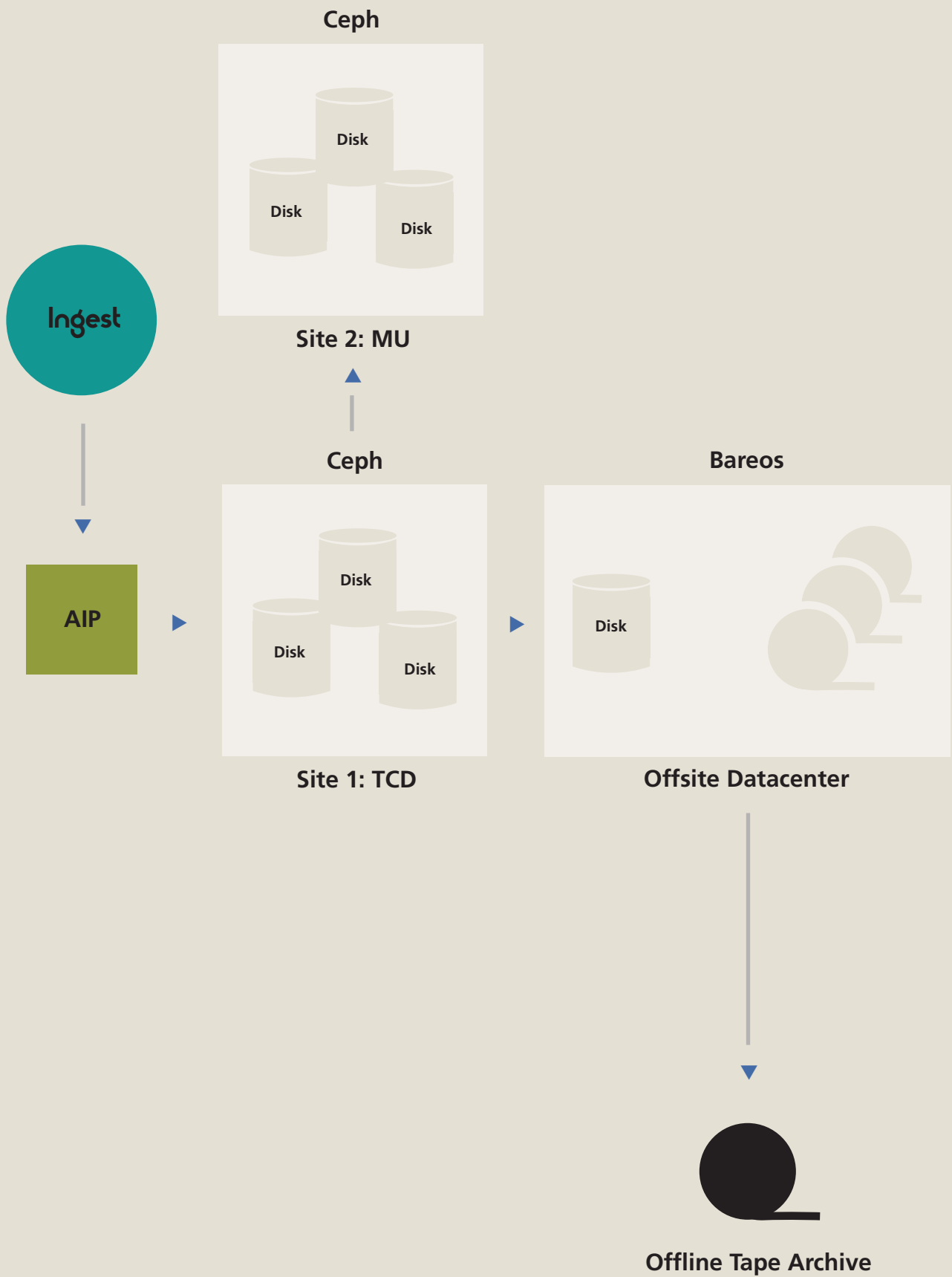
³⁹ <http://www.bareos.org/en>, last accessed 6 June 2015.

⁴⁰ <http://ceph.com/docs/master/rbd/rbd/>, last accessed 6 June 2015.

⁴¹ <http://ceph.com/docs/master/cephfs/>, last accessed 6 June 2015.

⁴² <http://ceph.com/docs/master/radosgw/>, last accessed 6 June 2015.

Figure 5.2: Archival storage workflow



As a function of preservation, the archival object workflow is the most important aspect of the Repository's storage solution. Data is first saved to disk in site one (TCD), it is then replicated to site two (MU) before being backed up offsite, first to disk and then to tape. On ingest the Archival Information Package (AIP) is created and stored in a separate archive pool over CephFS. The AIP is "an information package that is used to transmit archival objects into a digital archival system, store the objects within the system, and transmit objects from the system" (Consultative Committee for Space Data Systems (CCSDS), 2012). This is then replicated to a second, geographically separate, data centre over secure channels. Regularly, using Bareos, this is backed up to cold storage located in a third geographically separate data centre; first to disk and then to tape.

The AIP is structured on the MOAB format (Anderson, 2013). This format contains versioning, full metadata and checksum manifests. The benefit of versioning at the application layer, as opposed to lower down in the storage layer, is simplified storage management and layout, efficiency of tape archival, reduction or elimination of duplication in storage, visibility of versions to the user and ease of recovery in the event of asset integrity failure.

5.3. Storage integrity and trust

Ceph has very high levels of data protection. It has its own in-built checksumming and regularly checks and automatically fixes any errors it finds in data. It is considered "self-healing"⁴³. Bareos also keeps checksums and upon backing up of digital objects, will run checks to see if any errors have occurred, either on disk or on tapes.

A separate process, independent to Ceph and Bareos, checks the integrity of stored archival objects. This process opens the AIP, recalculates checksums and compares against the AIP manifest. It does this for all replicas of the AIP.

5.4. Storage migration

Migration of the entire storage solution could become necessary if the system reaches the limits of its scale, or the end of its supported life (for hardware or software components). In such a case it would be necessary to move the data to an upgraded or other external system. A data migration plan was developed to address this. The plan outlines the steps required to move data as seamlessly as possible off the current system. Data consistency checks are part of this process. As the system uses open standards and protocols, the storage layer can be changed or replaced without affecting the software stack. This allows for easier transfer of data to another storage solution should the need arise. Migration to commercial storage solutions is also possible.

⁴³ <http://ceph.com/docs/master/architecture/#rebalancing>, last accessed 6 June 2015.

6. Metadata and Data Modelling

Another important aspect in achieving the DRI's goal of building a TDR is that the underlying data management layer must be capable of dealing with a wide variety of digital collections. At the same time the data management layer requires a set of data models that allow preserved data and metadata to be accessed and analysed by new tools developed in the future. Supporting such a diversity of collections is challenging; individual collections are prepared and catalogued differently using metadata standards most suitable for that type of collection. Some of the most commonly used standards include Dublin Core⁴⁴ (DC) for academic, scholarly digital collections; Encoded Archival Description⁴⁵ (EAD) for archives; and Machine-Readable Cataloging⁴⁶ (MARC), and Metadata Object Description Schema⁴⁷ (MODS) for libraries (O'Carroll and Webb, 2012).

The use of appropriate data documentation frameworks (metadata standards) is key to data dissemination, and also impacts discoverability, search and dissemination of datasets. It is also important that a repository's data model captures and represents rich contextual information to enable effective data discovery, sharing and reuse. The decisions on how data and relationships are represented in a digital repository are critical, and also influence how users interact with the repository.

In this respect, two key requirements have driven the core design of the data management layer. The first requirement was to support the most commonly used data, and metadata, formats and standards. The second key requirement was to support effective search functionality across collections. In this respect, the DRI data models perform metadata and data indexing, with the support of Apache Solr, as discussed in Section 4.3. Moreover, the data models also incorporate data relationships management to allow for a meaningful presentation of data to the end user and enhanced data with contextual and linking information, to facilitate browsing across collections.

6.1. Data models overview

Data and metadata are stored in the underlying digital repository as collections of Fedora digital objects (Lagoze et al., 2005) that can be accessed by Hydra. As the user interface is a customised Hydra head, and to facilitate the creation, management and storage of digital objects, the data models are implemented as a Ruby gem, that makes use of ActiveFedora⁴⁸, a Ruby gem for creating and managing objects in the Fedora Repository

⁴⁴ <http://dublincore.org/documents/dces/>, last accessed 6 June 2015.

⁴⁵ <http://www.loc.gov/ead/>, last accessed 6 June 2015.

⁴⁶ <http://www.loc.gov/marc/>, last accessed 6 June 2015.

⁴⁷ <http://www.loc.gov/standards/mods/>, last accessed 6 June 2015.

⁴⁸ https://github.com/projecthydra/active_fedora, last accessed 6 June 2015.

Architecture (Project Hydra, 2009). Implementing the data models as extensions of ActiveFedora presents numerous advantages. Firstly, it provides a robust framework for working with Fedora digital objects, negating the need to implement such management functionalities from scratch. Secondly, ActiveFedora makes use of two key components that provide both XML-based metadata management and integration with Apache Solr. The first of these components is the Opinionated Metadata (OM) Ruby gem (Project Hydra, 2010a); the second is the Solrizer gem (Project Hydra, 2010b). These are both described in more detail below.

Each of the supported metadata standards are implemented as a set of classes which extend from the `ActiveFedora::Base` class. This allows modelling of Fedora digital objects as Ruby classes that incorporate a set of Fedora datastreams to handle the different types of metadata required by the system. Examples of metadata types include descriptive metadata, technical metadata and DRI administrative metadata, which incorporates preservation metadata.

ActiveFedora also facilitates the specification of relationships between digital objects as Rails associations (Project Hydra, 2012). These relationships are saved using the RDF (Resource Description Framework⁴⁹) specification in a special datastream, RELS-EXT. This allows for easier access and management of the relationships between stored digital objects, as well as exposing this relational information to third party applications. Moreover, it sets the basis for future extensions of the data models, for example supporting Linked Data.

Figure 6.2 provides an overview of the data models' classes, which can be summarised as follows:

- Digital Object classes for the creation, management and storage of different types of digital objects, based on their source metadata.
- GenericFile classes implementing functionalities for the management of the physical assets associated with metadata digital objects.
- Metadata Datastream classes for managing XML-encoded metadata.

The data models also include a set of Ruby modules and 'mixins' which implement attributes and methods that are common to the different data models classes. For clarity, these have not been included in the diagram.

⁴⁹ <http://www.w3.org/RDF/>, last accessed 6 June 2015.

Figure 6.1: Overview of the DRI architecture with data models highlighted

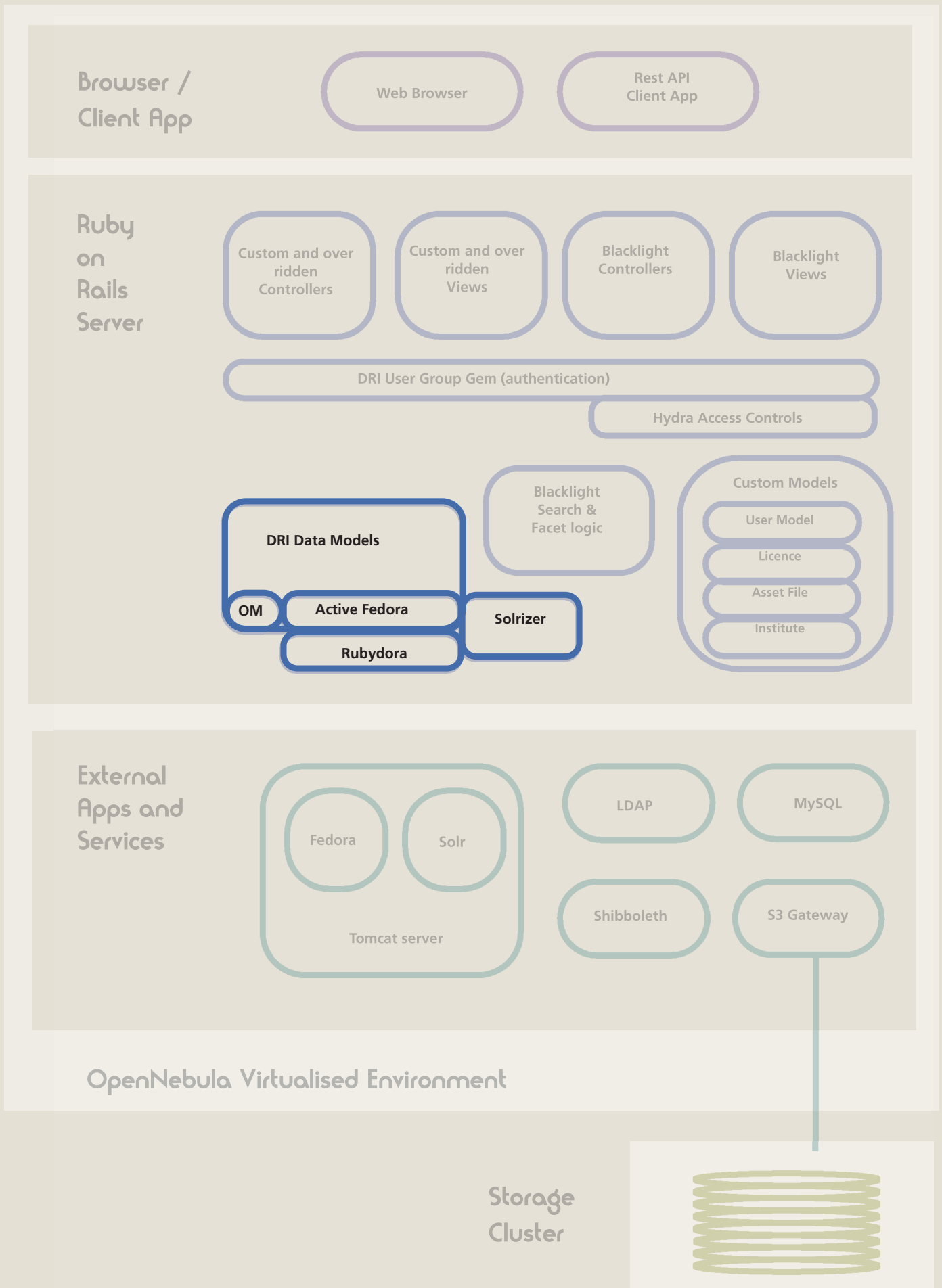
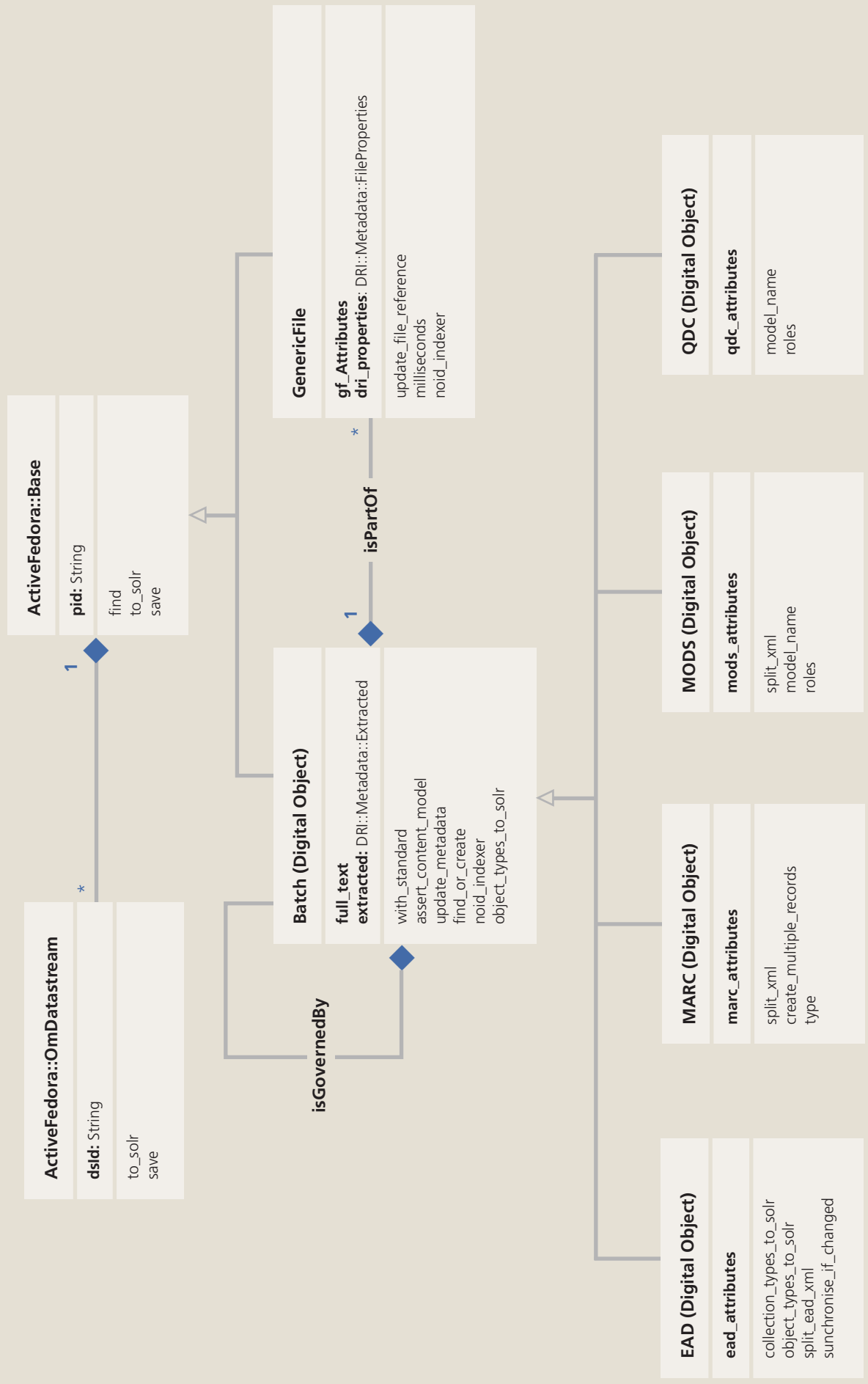


Figure 6.2: Overview of the Data Models (class diagram)



The ways in which the data models manage different kinds of collections, based on their source metadata, follow principles and approaches similar to other archives and libraries, for example Atrium's EAD content modelling for image-based collections (Johnson, 2011). Unlike the majority of repository implementations, which only support one metadata format, the DRI data models include support for multiple metadata standards, as well as a set of common metadata terms. These common metadata terms are DRI specific terms, for example title, creator and date, that allow for cross-collection search as well as visualisation tools.

Adding support for multiple metadata standards to the data models has been challenging for a number of reasons. The main challenge is the difference between the structures of the four standards chosen. For example, DC collections present a flat structure; there is only one collection container which holds all of the sets of metadata, data and digital objects. In contrast, EAD collections present a complex, hierarchical structure that allows for the definition of a large number of hierarchy levels, which requires the modelling of additional types of objects.

In terms of the data modelling or data representation, the approach taken was based on the design approaches most commonly used within the Hydra community. For example, digital resources within collections are represented as sets of different types of digital objects: digital objects containing descriptive metadata about the resource are linked to digital objects (Generic Files) that store technical metadata and the location of any digital assets associated with the digital resource (example of such assets include images, documents, etc.). This allows for the implementation of a set of data models for the management of "Hydra-compliant" digital objects (Duraspace, 2012). Hydra-compliant objects held in Fedora are expected, by Fedora, to have a DC metadata datastream, and a RELS-EXT datastream which declares one or more appropriate content model (cModels) for the digital object to subscribe to, as well as relationship information.

Additionally, Hydra requires these objects to have an enforceable rights statement. This is either in a rightsMetadata datastream, which is currently the most common pattern, and/or an Admin Policy Object⁵⁰ (APO) which governs the digital object. The data models implement rightsMetadata datastreams for managing digital object permissions as well as access policies to enable any form of access/delivery of the digital objects to the user. The management of access or rights metadata is implemented in a separate Ruby Gem called UserGroup, developed by the team. In addition to these required datastreams, the Repository incorporates a number of additional datastreams which vary depending on the content type of the digital object. These are described in detail in the Section 6.2.

⁵⁰ <https://github.com/projecthydra/hydra-head/wiki/Access-Controls>, last accessed 6 June 2015.

6.2. Data management: from XML to repository modelled data

The Repository currently supports a number of mechanisms for ingesting data and the accompanying metadata into the repository. Digital objects can be ingested through third-party applications via the DRI REST API (see Section 3.2). They can be ingested through a web form based ingest via the user interface. Finally they can be ingested via the Client application which, although it is currently command line based, will incorporate a graphical user interface.

Once metadata are uploaded, a number of validations are performed to ensure validity and correctness, prior to ingest. As the metadata are XML-encoded, the first validation involves checking whether the source XML is well-formed and conforms to its associated metadata schema. For DC, MODS, and MARCXML metadata validation against XML Schema Definition (XSD) is performed. For EAD either Document Type Definition (DTD) or XSD validation can be performed depending on the format of the source metadata.

Additional DRI-specific validations are also required, for example ensuring that the provided metadata include the mandatory DRI fields, and performing data type based checks. Temporal metadata fields, including dates or date ranges, are checked to ensure that they are encoded using the recommended standards for temporal data representation, ISO 8601⁵¹ or W3C w3cdtf⁵².

Valid data and metadata can then be processed and stored in the Repository using the data models. The business logic of the data management layer can be divided into two main controllers. The first controller handles the structure, storage, and retrieval of metadata, while the second applies the same processes to the digital assets, i.e. the data itself.

6.2.1. Metadata management

As introduced earlier (see Section 6.1), the data models follow the design principles for Hydra-compliant digital objects where every digital object must incorporate a minimum set of datastreams. The data models incorporate a number of additional datastreams for metadata management. As collections may present different internal structures (flat versus hierarchical, depending on their associated metadata standard), the type and number of these datastreams vary.

⁵¹ <http://www.iso.org/iso/home/standards/iso8601.htm>, last accessed 6 June 2015.

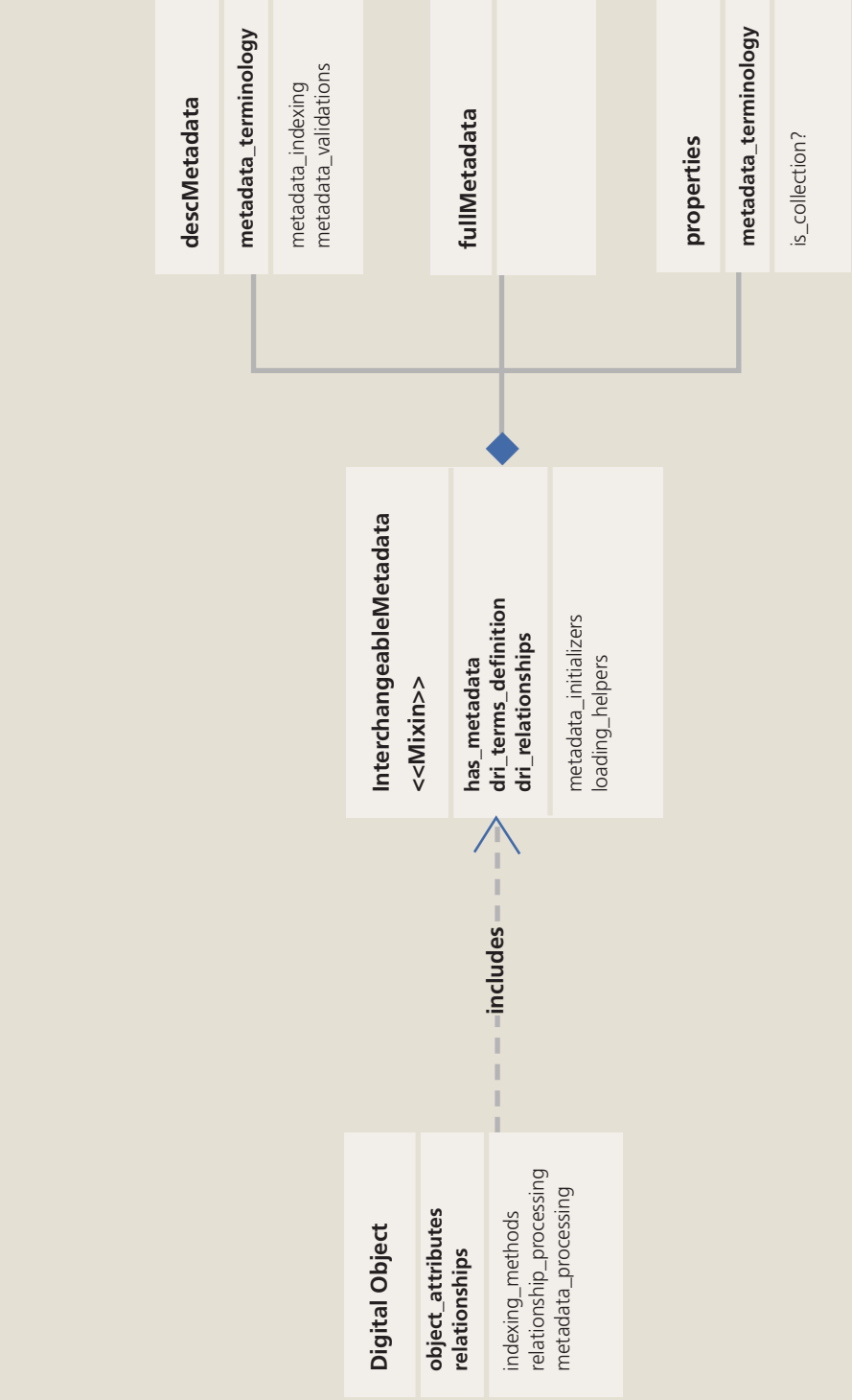
⁵² <http://www.w3.org/TR/NOTE-datetime>, last accessed 6 June 2015.

For example, collections encoded using a flat metadata structure, such as Dublin Core, are represented as sets of digital objects (the metadata objects) which are governed by 'container' digital objects (collection/sub-collection objects). Each of the digital objects, one per metadata record, can be associated with other generic file objects which represent physical assets in the Repository. For this kind of collection, only one type of metadata datastream (DC descriptive metadata datastream) is required as the only descriptive difference between metadata objects and container objects is their type.

In contrast, for collections with a hierarchical structure, such as EAD or MODS, each of the components within the hierarchy are described in the XML using different metadata terms. For example, in EAD the collection container is described using the <archdesc> element, and any components within the collection are described using the <c> element. In terms of the data models, this means that different types of datastreams are needed to store the XML that describes the different types of components, so as to distinguish collection objects from sub-collections or bottom-level objects within the hierarchy.

In Hydra and Fedora terms, each metadata record is modelled as a compound object with two datastreams storing the XML-encoded metadata. This information is stored in the form of XML snippets. Figure 6.3 shows the different types of datastreams for metadata objects. The first type of datastream, descMetadata (descriptive metadata), stores metadata at the object level. For digital objects in a collection with a hierarchical structure, only the metadata associated with an object is stored in this datastream. Metadata objects within the hierarchy also store, in the second type of datastream (fullMetadata), a more in-context XML snippet with information about the object's immediate children. Storing metadata information with such a low level of granularity allows for indexing a collection's individual parts into Solr so they can be independently discovered and visualised. At the same time, they contain sufficient contextual information so they can refer back to other relevant objects within the collection.

Figure 6.3: Class diagram showing metadata digital object and its associated datastreams



The information stored in the metadata datastreams are managed by Ruby classes which extend from `ActiveFedora::OmDatastream`. `OmDatastream` is a Ruby class included in the Opinionated Metadata (OM) gem⁵³ (see section 6.1). Using this class, the DRI data models can easily manage Fedora XML datastreams since it allows for assigning Ruby attribute accessors (with Xpath support) to specific XML elements. In this way, metadata information can be easily managed without having to access the source XML directly.

ActiveFedora also provides attribute accessors to datastreams extending from `OmDatastream`, so that objects at higher levels within the data model have direct access to the metadata terms stored in datastreams. For example, the data model's 'title' accessor provides access to the information contained in the metadata 'title' term. Calling this accessor will invoke the title accessor defined in `OmDatastream`, which in turn allows for accessing the source XML where the title information is actually stored. Another advantage of `OmDatastream` is its native support for Solr indexing. ActiveFedora includes mechanisms to convert between the standard Ruby hash data structure and a Solr document data structure, i.e. the ActiveFedora model implements methods that transform metadata information into a data structure in which each data is stored in key-value pairs, with each key being the name of a Solr field, and each value holding the data to be indexed.

6.2.2. Digital assets management

Uploaded preservation-quality assets undergo a rigorous set of checks before they are permanently stored in the Repository. As set out by requirements, incoming files are scanned for malware and validated for file type correctness. Failure of the malware check forces the Repository to reject the uploaded files, while other tests deliver error messages to the user attempting the upload. If all tests are passed, the assets are preserved in the internal storage system (see Section 5.2).

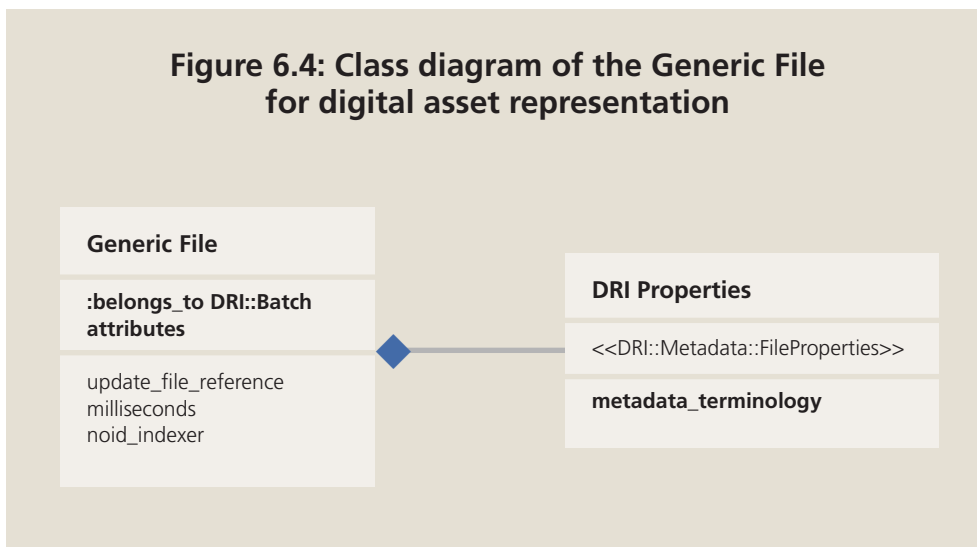
Similar to how metadata digital objects are treated, a digital asset is modelled as a class extending from `ActiveFedora::Base` (i.e. `DRI::GenericFile` introduced above). Generic files are digital objects with content-bearing datastreams to represent the physical contents of the digital asset, as well as a datastream that contains descriptive information (technical metadata) about the asset. Figure 6.4 shows the classes that have been implemented to represent Generic Files.

In terms of modelling this kind of digital object, Hydra models typically follow two different approaches ("Design principles," n.d.). The first is based on the definition of

⁵³ <https://github.com/projecthydra/om>, last accessed 6 June 2015.

'compound' content objects with a number of content-bearing datastreams, whilst the second approach defines 'atomistic' or complex objects where a parent object is linked to content held in one or more child objects. In the case where compound objects have just one content-bearing datastream they are usually referred to as 'simple' content objects. Compound objects can be useful in terms of dissemination, as one digital object can store preservation-quality assets, as well as lower resolution assets for internet delivery. However, the DRI data models implement 'simple' content objects: the preservation-quality asset being represented in the content-bearing datastream. Lower resolution files can then be generated for dissemination via the Repository's user interface. The rationale for this is that it translates into storing less complex digital objects, for which preserved materials are separate from purposively generated ones which, if needed, can be updated and/or regenerated.

Figure 6.4: Class diagram of the Generic File for digital asset representation



Content digital objects are stored in the digital repository once they have passed all the system tests and Fedora digital objects are then generated. A one-to-many association links the Generic File model used to represent assets with its associated metadata digital object. This creates a simple child-to-parent data model association (isPartOf) that links many assets to one metadata digital object.

Once the asset is saved in Fedora, it is added to the Repository's surrogate Resque⁵⁴ queue, where uploaded preservation-quality media files are converted into lower-quality surrogate files intended for access and discovery. All surrogates are automatically generated by background processes running in the servers. Maintaining a centralised system

⁵⁴ <https://github.com/resque/resque>, last accessed 6 June 2015.

for generating surrogates is resource intensive but ensures full control of surrogate generation. Furthermore, it makes the adoption of future internet technologies a seamless process. It will allow the DRI to easily update the parameters needed for the creation of new types of surrogates, as well as to meet future end users' expectations of how these media files are displayed.

Within the Repository, the preserved assets are treated in a similar fashion to preserved metadata records. The goal is to extract as much descriptive information as possible from asset contents, while at the same time, making this information available to users. The Repository extracts machine-readable file information such as the file's dimensional characteristics, statistical information and metadata automatically generated by source tools. Currently, the Repository uses Harvard's FITS (Harvard, 2009) tool to extract these metadata fields, storing the results as a FITS XML file. As well as the metadata fields, the Repository also extracts full-text from textual documents, which allows users to perform complex searches on a document's full text. In this respect, Apache Solr provides a built-in functionality for the indexing of full-text, and also supports a wide variety of document file types. For this, Solr Cell is used as it facilitates indexing all document types supported by the Apache Tika⁵⁵ project.

6.3. Data indexing, search and retrieval

The Repository's data models handle indexing into Solr, which is performed using ActiveFedora's built-in indexing support, through the Solrizer Ruby gem. Both digital objects and datastreams extending from ActiveFedora have access to Solrizer indexing methods, specifically, to the `to_solr` method. Invoking this method from a digital object's datastream triggers the transformation of metadata contents into a Solr document. This is an XML document in Solr's expected index format. Solrizer then commits the index document into the Solr instance. For digital objects, the process is repeated for each of the datastreams of the digital object. Saving digital objects triggers `to_solr` calls, which in turn send the contents to be indexed into Solr, thus allowing for automatic updates. The process of indexing is illustrated in Figure 6.5.

The data management layer uses indexing to present the data it stores in meaningful ways to the end user. The Repository's user interface provides the user with searching and faceting or filtering functionalities, with the support of Apache Solr search engine. This allows the user to focus their search results from the large number of available datasets, as well as to explore the contents of collections in a more user-friendly and interactive way. Faceting breaks up search results into multiple categories and allows

⁵⁵ <http://tika.apache.org/>, last accessed 6 June 2015.

Figure 6.5: Sequence diagram showing the process of indexing into Solr



the user to further restrict search results, based on the facets selected.

The source metadata format used does not restrict indexing into Solr as a result of the data models' transformations performed on ingest. One such transformation would be the processing of personal names. For example, searching for "Eamon De Valera" as an author will return objects that store any version of that name in the <origination> (EAD) and <creator> (Dublin Core) metadata fields.

Similar processes also apply to the presentation of temporal data. An example of this is how the data models manage date ranges stored as DCMI periods: e.g. the value "1721-1730", stored in the <date> field. Rather than simply storing it as two dates, as many systems would automatically do, thereby missing the meaning of the range as understood by a human reader, the data models make use of Solr indices that are specifically designed to handle dates and date ranges. When searching the Repository this allows for the return of any digital object that falls within the specified date range.

The rationale for such transformations to be applied to the original, source metadata is to enhance how data is presented to the end user. The Repository's indexing and faceting allows the linking of data through meaning; the information is presented and searched in a semantic way, rather than treating it as plain text. This indexing information is available to third party tools through the API.

It is also possible for third party developers to create their own data model to interpret the Repository's stored data in order to expand on this semantic interpretation, or to present to the end user the stored data in a different format.

6.4. Relationship representation in the data models

In addition to data indexing, the ability to represent and store data relationships can enhance data visualisation and also provide the end user with richer contextual information. This is particularly important when exploring a vast number of collections. The data models incorporate data relationships management when representing collections in the Repository. Figure 6.6 shows how relationships can be used to represent the structure of a collection stored in the digital repository. The particular choice of supported relationships was selected based on the needs of the individual the demonstrator projects, as well as on their degree of appropriateness for describing the kinds of collections that will be held in the Repository. The relationships supported by the data models can be grouped into two main categories:

- Data models internal relationships, which are mostly used for representing collection structure and membership
- Metadata-defined relationships, which are used for representing associations between digital objects

The first type of relationship is common to every object, independent of its metadata type, and can be used to express navigation or hierarchy. Examples of this type of relationship include: `isGovernedBy`, `isMemberOfCollection/hasMember`, or `isPart/hasPart`. The second type of relationship is provided via the descriptive metadata associated with digital objects. The data models implement a different set of relationships for each supported metadata standard as the type and number of relationships vary from one standard to another. Examples of this type of relationship include `isPrecededBy/`
`isSucceededBy`, `isDocumentationFor`, and `isReferencedBy`.

Figure 6.6: Example of relationship representation for DRI collections

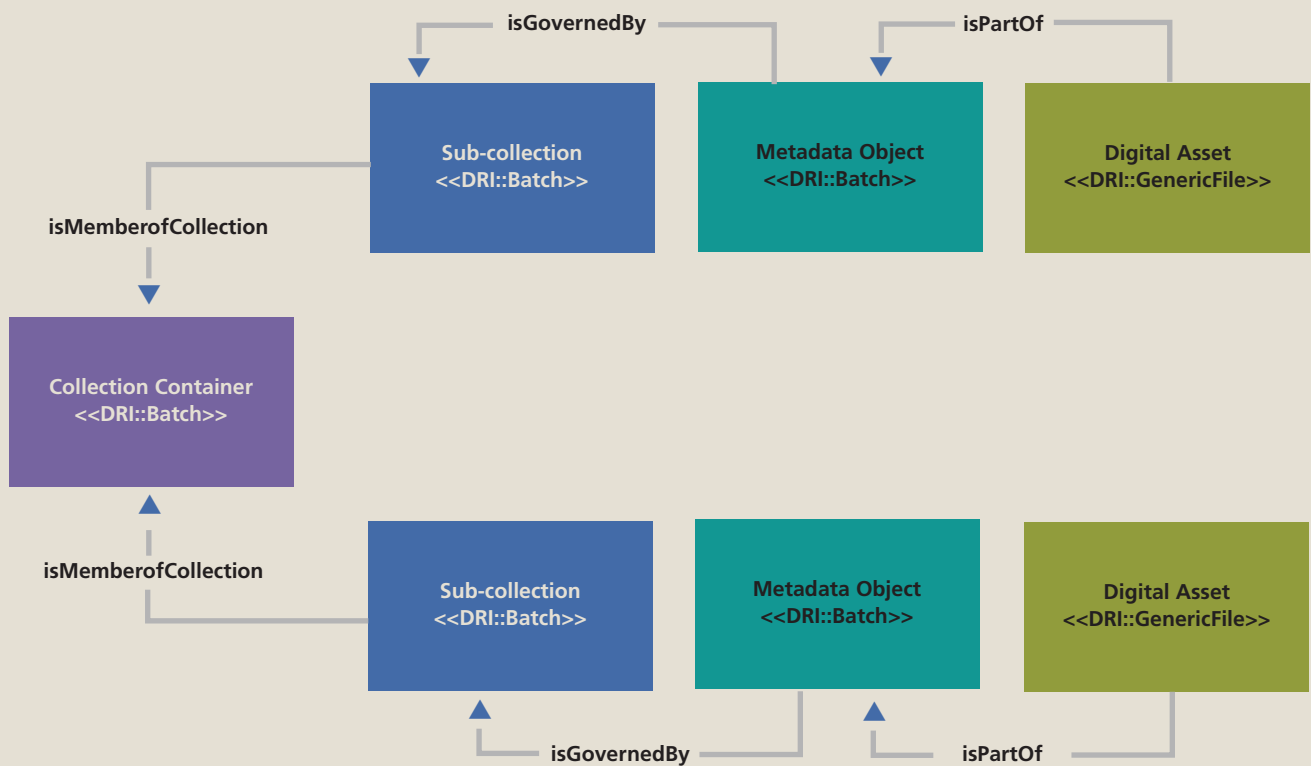


Table 6.1: Supported relationships in DRI

Metadata Standard	Relationship Name	Relationship	Note
DC	Relation	dcterms:relation	Link to internal DRI objects, or external digital materials
	Is Part Of	dcterms:isPartOf	Collection structure/hierarchy representation
	Is Referenced By	dcterms:isReferencedBy	Link to internal DRI objects, or external digital materials
	References	dcterms:references	Link to internal DRI objects, or external digital materials
	Is Version Of	dcterms:isVersionOf	Link to internal DRI objects, or external digital materials
	Has Version	dcterms:hasVersion	Link to internal DRI objects, or external digital materials
	Is Format Of	dcterms:isFormatOf	Link to internal DRI objects, or external digital materials
MODS	Preceding	relatedPreceding	Expression of sequencing
	Succeeding	relatedSucceeding	Expression of sequencing
	Host	relatedHost	Collection structure/hierarchy representation
	Constituent	relatedConstituent	Collection structure/hierarchy representation
	Original	relatedOriginal	Link to internal DRI objects, or external digital materials
	Other Version	relatedVersion	Link to internal DRI objects, or external digital materials
	Review Of	relatedReview	Link to internal DRI objects, or external digital materials
	Other Format	relatedFormat	Link to internal DRI objects, or external digital materials
	Is Referenced By	relatedReferencedBy	Link to internal DRI objects, or external digital materials
	References	relatedReferences	Link to internal DRI objects, or external digital materials
	Series	relatedSeries	Link to internal DRI objects, or external digital materials
MARC	Other Edition Entry	MARC Tag: 775	Link to internal DRI objects, or external digital materials
	Additional Physical Form Entry	MARC Tag: 776	Link to internal DRI objects, or external digital materials
	Preceding Entry	MARC Tag: 780	Expression of sequencing
	Succeeding Entry	MARC Tag: 785	Expression of sequencing
	Other Relationship Entry	MARC Tag: 787	Link to internal DRI objects, or external digital materials

Table 6.1 summarises the relationships that have been implemented in the data models, for each of the supported metadata standards.

From a technical perspective, relationships are implemented in the data models by using ActiveFedora's built-in support for relationships definition. Relationships are represented as Rails associations between objects (Coyne, 2013), and the associations used are of the types 'has_many', a one-to-many relationship, and 'belongs_to', a one-to-one relationship. Additionally, ActiveFedora uses RDF to represent these relationships. Whenever an association between two objects is created, ActiveFedora writes this metadata into the RELS-EXT datastream of the related digital objects. The sample code below shows how ActiveFedora relationships are defined in Ruby, as well as their translation into RDF assertions in the RELS-EXT datastream.

Figure 6.7: ActiveFedora relationships as defined in Ruby

```
class Batch < ActiveFedora::Base
  has_many :parts, :property => :is_part_of, :class_name => `DRI::Batch`
  belongs_to :is_governed_by, :property => :is_part_of, :class_name =>
`DRI::Batch`
  ...
end
-----
-
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDFxmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ns0="info:fedora/fedora-system:def/model#"
xmlns:ns1="info:fedora/fedora-system:def/relations-external#">
  <rdf:Description rdf:about="info:fedora/pid:2">
    <ns0:hasModel rdf:resource="info:fedora/afmodel:DRI_Batch"/>
    <ns1:isPartOf rdf:resource="info:fedora/pid:1"/>
  </rdf:Description>
</rdf:RDF>
```

The model's RDF assertions shown above are automatically created based on the data model name from the ActiveFedora model. The rest of defined relationships take their name from a configuration file (`predicate_mappings.yml`) that ActiveFedora uses to look up RDF predicates, based on the property name being specified in the data model. The data models override this file to include a custom set of specific relationships, as well as the set of relationships implemented for each of the supported metadata standards.

Currently the data models processes relationships if the information is included in the sourced metadata, but in the future will support their addition via the user interface. Providing support for adding relational information through the user interface is highly important as some of the supported metadata standards, for example MARC, present limitations in how relationship information can be specified in the metadata.

7. User Interface Design

As part of the overall requirements gathering process (see Chapter 2) and to provide a brief for later visual design phases, an evaluation of information access use cases was carried out. This involved examining real-world digital libraries, repositories and other non-academic digital asset management sites. Each was considered from the point of view of functionality, usability and accessibility.

Following the evaluation of user interfaces a number of recommendations were made regarding the architecture of the user interface (UI). The Repository should:

- Make search integral to the interface.
- Support faceted browsing capabilities across collections and broad categories.
- Develop the interfaces within a well-supported, Open Source, framework and utilise well-supported standards based visualisation tools.
- Provide object usage and geolocation visualisations.
- Provide timeline visualisation.

7.1. Interface design principles

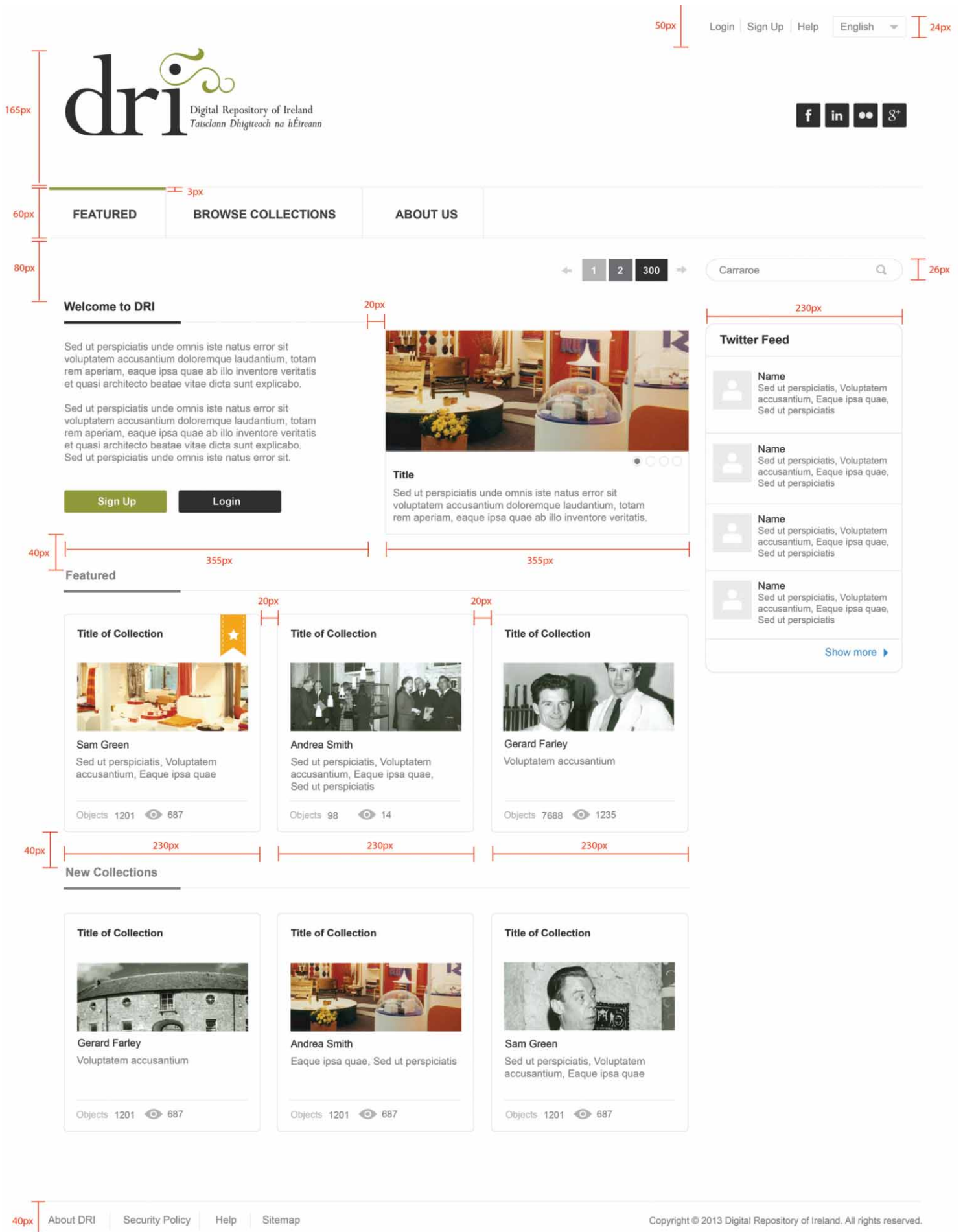
It is important to note that a decision was made in the proposal stages of the project to only support HTML5⁵⁶ compliant web browsers. HTML5 is the W3C's candidate recommendation for an fully interoperable HTML standard since July 2013. This has allowed the development team to concentrate on modern techniques and supporting software frameworks without the need to consider support for legacy browsers.

As discussed by Johnson (2010) the use of grid-based design systems predates the Web. In Johnson's view grid designs are simply adhering to the universal design principle of alignment. Grid designs simplify the information retrieval task for users by making the display readily understandable. The more rapidly users can identify a pattern to the information the more quickly they can move on to information analysis.

With these design principles in mind the DRI desktop web layout is based on a four column 1024px fixed grid. This layout allows for a balance between a predictable desktop layout and an uncomplicated reconfiguration of the display for mobile devices based on a 230px width.

⁵⁶ <http://www.w3.org/TR/html5/>, last accessed 6 June 2015.

Figure 7.1: DRI desktop fixed grid concept design




CSS Media queries were introduced in the W3C CSS3 recommended standard in June 2012. They allow the browser to select a CSS style based on the capabilities of the device displaying it. The most important of these queries is the width of the browser. The designs also make use of semantic layout, including the use of meaningful container elements such as header, article, aside, nav and footer to aid accessibility.

Adaptive and responsive web design is a relatively recent web design approach. This approach acknowledges the range of device and use contexts which web sites must now provide for. New device contexts such as tablet computers and smartphones now account for a substantial proportion of all web site usage (Google, 2015) and the Repository is expected to be accessed across all device types.

There are two similar but subtly different possible approaches to dealing with device capabilities in CSS. The first, responsive design, uses fluid designs to allow the page layout to change incrementally across a range of screen sizes using fluid percent range based CSS markup. Adaptive design by contrast sets 'breakpoint' values for screen size that changes the layout (sometimes dramatically) when, for example, a browser screen below a certain value is detected. The overall aim of both approaches is to provide the user with an experience which is optimal for their use context.

The design takes an adaptive approach that allows the desktop media rules to present a known layout that is of fixed width. The tablet and smartphone use context are given separate CSS while making use of the same HTML markup. This overall approach makes the redesign process relatively straightforward and provides for a very clean separation of content and presentation.

Figure 7.2: Designs for tablet

 Digital Repository of Ireland
Léaráin Digiúcháin na hÉireann

[Admin](#) [Log Out](#) [English](#)

[f](#) [in](#) [t](#) [t](#)


Search...

[HOME](#) [DISCOVER](#) [ORGANISATIONS](#) [ABOUT](#) [WORKSPACE](#) [LOG OUT](#)

Newest Items per page 9 Save Current Search

Collections + **Objects** 1 | 2 | ⇨


Child's dancing costume.



A Sense of Identity | Manners and customs | Dance | Handicraft |

[Image](#)


Tara brooch.



A Sense of Identity | Handicraft | Design | Decorative arts | Metalwork |

[Image](#)


Starry plough flag.



A Sense of Identity | History | Irish Citizen Army | Easter Rising, 1916 |

[Image](#)


Hibernia awakens from her slumbers.



A Sense of Identity | Handicraft | Ceramics | Pottery | History |

[Image](#)


Clare Kennedy wearing Celtic Revival costume.



A Sense of Identity | Headbands | Manners and customs | Clothing and

[Image](#)


Celtic Revival sash and head-band.



A Sense of Identity | Manners and customs | Clothing and dress |

[Image](#)


Daniel O'Connell chess set.



A Sense of Freedom | O'Connell, Daniel, 1775-1847 | Politicians |

[Image](#)


Daniel O'Connell commemorative medal.



A Sense of Freedom | O'Connell, Daniel, 1775-1847 | Politicians |

[Image](#)

Daniel O'Connell jug.




A Sense of Freedom | O'Connell, Daniel, 1775-1847 | Politicians |

[Image](#)

1 | 2 | ⇨

Figure 7.3: Design for search result page – desktop version



[Admin](#) | [Log Out](#) | [English](#)


f in t t

HOME
DISCOVER
ORGANISATIONS
ABOUT
WORKSPACE ▾
MANAGE ▾
LOG OUT

[Save Current Search](#)

Collections + Objects

1 2 →




Aran style woollen jumper.

This Aran jumper was given to the National Museum by Rualrí Ó Siocháin, the son of entrepreneur Pádraig Ó Siocháin. Through his company, Galway Bay Products, he gave

A Sense of Place Manners and customs Handicraft Clothing and dress Sweaters Aran

[Image](#)




Celtic Revival sash and head-band.

This sash style belt and head-band are parts of a Celtic Revival costume made and worn in the 1920s. The full costume is composed of a full length dress, sash-belt, veil, head-band

A Sense of Identity Manners and customs Clothing and dress Handicraft Embroidery

[Image](#)




Child's dancing costume.

Child's dancing costume in green fabric, embroidered with Celtic motif, with crochet collar and cuffs. Cape lined with orange fabric and white belt complete the colours of Ireland as

A Sense of Identity Manners and customs Dance Handicraft Design Decorative arts

[Image](#)




Clare Kennedy wearing Celtic Revival costume.

This photograph shows Claire Kennedy in a Celtic Revival costume made and worn in the 1920s. The full costume is composed of full length dress, sash-belt, veil, head-band and

A Sense of Identity Headbands Manners and customs Clothing and dress Handicraft

[Image](#)




Daniel O'Connell chess set.

Chess set, reputed to have been used by Daniel O'Connell and his friend, fellow lawyer and politician Richard Lalor Sheil while O'Connell was imprisoned in Richmond Gaol in

A Sense of Freedom O'Connell, Daniel, 1775-1847 Politicians History Catholic

[Image](#)




Daniel O'Connell commemorative medal.

Commemorative medal marking the death of Daniel O'Connell. This is one of a variety of objects used to mark the death of Daniel O'Connell and celebrates his campaign for

A Sense of Freedom O'Connell, Daniel, 1775-1847 Politicians History Catholic

[Image](#)




Daniel O'Connell jug.

White jug, with printed image commemorating Daniel O'Connell's election as M.P. for Co. Clare in the By-Election of 1828 for the parliament based at Westminster in London. One

A Sense of Freedom O'Connell, Daniel, 1775-1847 Politicians History Catholic

[Image](#)




Daniel O'Connell's 'Repeal' cap.

Daniel O'Connell's 'Repeal' Cap, made of green velvet and embroidered with gold shamrocks, with a gold lace tassel and button on top. The button bears a crowned harp

A Sense of Freedom O'Connell, Daniel, 1775-1847 Politicians History Catholic

[Image](#)



Hair hurling ball.

Hurling ball made from matted cow hair with a plaited horsehair covering radio carbon dated to the late fifteenth century. This hurling ball was found in a bog near Sneeem, Co

A Sense of Place Sports Games Gaelic games Hurling (Games) Archaeology Slottars

[Image](#)

Refine your search 14

Subject (Temporal) ▾

From

To

[Set](#)

Subjects >

Places >

Names >

Language >


Mediatype >

Collection >


1 2 →

[End User Agreement](#) | [About](#)


Digital Repository of Ireland
 Royal Irish Academy
 19 Dawson Street
 Dublin 2
 Tel +353 1 609 0674
 Email info@dri.ie




Ireland's EU Structural Funds
 Programmes 2007 - 2013
 Co-funded by the Irish Government
 and the European Union



EUROPEAN REGIONAL
 DEVELOPMENT FUND



An Roinn Post, Fintar agus Nialláiríochta
 Department of Jobs, Enterprise and Innovation



HIGHER EDUCATION AUTHORITY
 AN ÚDARÁS UCHTAR-ÓIDEACHAIS

7.2. Public and discovery interfaces

Digital repositories can be used for many reasons, but arguably the most important set of use cases centre around accessing information. There are also many content-specific and goal-directed requirements that might lead a user to access a collection of digital information (Saracevic, 2000).

The designs have adopted the ‘search everywhere’ pattern as suggested by Hearst and others (Hearst 2009). It is intended that the user should be able to trigger a content search using a text string and that this search form should be available on all pages. The initial search trigger is a keyword or keyword-wildcard ‘*’ search. This brings the user into the search result page where teaser views of digital objects are displayed. After triggering a keyword search the user is able to make use of the facet filtering to narrow the search. The Repository makes use of the Blacklight software component (see Section 4.3) to facilitate both the keyword search and facet filtering.

7.3. UI architecture

The user interface designs are realised using several technologies that are outlined below.

7.3.1. Ruby on Rails views

As the Hydra platform was selected as the framework architecture for Repository, the implementation of its user interfaces utilises the Model-View-Controller pattern of its programming language Ruby on Rails (see Section 4.3). A view in the default configuration of Ruby on Rails is a type of template file (erb file) which mixes HTML markup and dynamically generated data from the application and renders the final webpage. A single view may also be split into several separate erb files or view “partials”. Typically these partials correspond to some functional part of the page such as the main content or navigation. It is by manipulating these view partials that the overall HTML rendering is achieved. Non-HTML views can also be created to allow rendering in different formats, such as XML or JSON ⁵⁶.

7.3.2. JavaScript framework

The JavaScript framework used by Repository is JQuery. It provides libraries and functions of common JavaScript tasks such as carrying out AJAX requests and performing UI animation. This has been the standard Ruby on Rails framework since version 3 and is used

⁵⁶ JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. <http://www.json.org/>, last accessed 6 June 2015.

commonly used by components such as Blacklight. Since the UI team was already familiar with JQuery it was decided to continue development using this framework, which is arguably becoming the de facto standard (W3Techs, 2014).

7.3.3. SASS and Grid framework

The complex adaptive layouts envisaged for the Repository makes the task of producing CSS quite difficult. To aid in the task the Repository makes use of the SASS language⁵⁷ and a grid framework called Zen Grids⁵⁸.

SASS is an extension of the CSS3 language that adds programmatic elements from other languages such as nested rules, variables, mixins (similar to functions, procedures, or methods in other languages) and selector inheritance. SASS is not currently understood by web browsers natively but is compiled into well-formatted CSS3 using either a command line tool or a web-framework plugin, available as standard in Rails 3.

The Zen Grids Framework is a set of SASS mixins that provide tools for making responsive or adaptive grid based layouts. This allows the grid design layouts to be more easily specified and separated from the design of the block level element designs. This had the added advantage that block level designs can be treated almost independently of page layout.

7.4. Visualisations

Three types of data visualisations are being supported within the application. These are full-text context visualisation, also known as 'hit highlighting', search result timelines and search result mapping.

7.4.1. Timelines

Timelines are supported by rendering a JSON formatted search result within a JavaScript HTML5 timeline 'widget'. The framework currently being used is called TimelineJS. TimelineJS is an open-source tool that renders the JSON as a HTML5 compatible interactive timeline.⁵⁹

⁵⁷ <http://sass-lang.com/>, last accessed 6 June 2015.

⁵⁸ <http://zengrids.com/>, last accessed 6 June 2015.

⁵⁹ <https://github.com/NUKnightLab/TimelineJS>, last accessed 6 June 2015.

Figure 7.4: Timeline visualisation

dri Digital Repository of Ireland
Taisclann Dhigiteach na hÉireann

Admin | Log Out | English

Search...

HOME | DISCOVER | ORGANISATIONS | ABOUT | WORKSPACE | MANAGE | LOG OUT

Newest | Items per page: 9 | Save Current Search

Collections + Objects

700
Tara brooch.

This brooch was found not in Tara but near the seashore at Bettystown, Co. Meath, in 1850. Its provenance was attributed to Tara, the

1600
Hair curling comb.

600 700 800 900 1000

← CURRENT SEARCH TIMELINE →

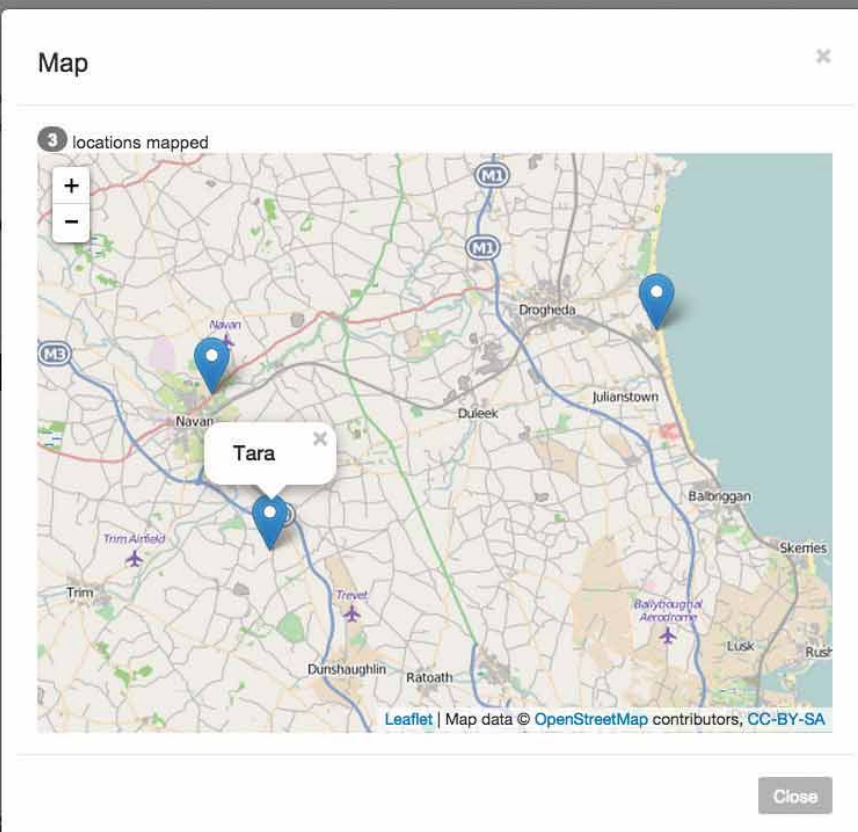
↑ ↓ 🔍

Tara brooch.

7.4.2. Mapping

Mapping of search results are also supported by rendering a JSON formatted result within another JavaScript framework. The framework currently being used is called OpenLayers⁶⁰. OpenLayers is a web mapping library which takes Geo data and using HTML5 techniques renders the data onto a map. Map tiles can be downloaded from a variety of both Open Source, free and commercial sources which gives the Repository good flexibility into the future.

Figure 7.5: Map visualisation



The screenshot shows a web application interface. On the left, there is a navigation menu with 'HOME', 'DISCOVER', and 'OR'. Below the menu, there is a search bar and a list of items, with 'Tara brooch.' highlighted. The main content area displays a map visualization. The map is titled 'Map' and shows three locations marked with blue pins: Navan, Tara, and Drogheda. The map is overlaid on a page for 'Tara brooch'. The map includes a search bar, a zoom in (+) and zoom out (-) button, and a 'Close' button. The map data is attributed to OpenStreetMap contributors, CC-BY-SA.

Title
Tara brooch.

Description
This brooch was found not in Tara but near the seashore at Bettystown, Co. Meath, in 1850. Its provenance was attributed to Tara, the hill of the ancient High Kings of Ireland to the west, by a dealer in order to increase its value. It is 18cm long (7 inches) made of cast and gilt silver and is elaborately decorated on both faces. The front is ornamented with a series of exceptionally fine gold filigree panels depicting animal and abstract motifs that are separated by studs of glass, enamel and amber. The back is flatter than the front, and the decoration is cast. The motifs consist of scrolls and triple spirals and recall La Tène decoration of the Iron Age. A silver chain made of twisted wire is attached to the

⁶⁰ <http://openlayers.org/en/v3.3.0/doc/>, last accessed 6 June 2015.

8. Conclusion

As discussed, the Repository development was based on a rigorous requirements gathering and management process which informed the overall development of the Repository by Strand 3, with input from the Work Packages, Task-forces and Working Group across Strands 1, 2 and 4. The development of the Repository will continue in the future; work on a Trusted Digital Repository is never complete. There are a number of further features planned including automated aggregation of (meta)data from partners, the expansion of preservation activities, support for additional metadata standards and a browser based bulk ingest tool. Over time hardware failures may occur, file formats become obsolete, user access devices and styles change and new tools and methods are developed. The DRI Repository must be able to respond to these new requirements over time and ensure the preservation and sustained access for the data held.

Bibliography

- Anderson, Richard. (2013). *The Moab Design for Digital Object Versioning*. Code4Lib Journal, (Issue 21). Retrieved from <http://journal.code4lib.org/articles/8482>
- Chelimsky, D., Astels, D., Helmkamp, B., & North, D. (2010). *The RSpec Book Behaviour Driven Development with Rspec, Cucumber, and Friends*. Pragmatic Bookshelf.
- Consultative Committee for Space Data Systems (CCSDS). (2012). *Reference Model for an Open Archival Information System (OAIS)*. Washington, D.C.: Magenta Book. Retrieved from <http://public.ccsds.org/publications/archive/650x0m2.pdf>
- Coyne, J. (2013). *Relationships*. Retrieved from https://github.com/projecthydra/active_fedora
- Data Seal of Approval Board. (2013). *Data Seal of Approval Guidelines version 2*. Retrieved from http://datasealofapproval.org/media/filer_public/2013/09/27/guidelines_2014-2015.pdf
- Davis, D., & Wilper, C. (2011). *Content Model Architecture*. DuraSpace wiki. Retrieved from <https://wiki.duraspace.org/display/FEDORA35/Content+Model+Architecture>
- Design principles. (n.d.). Retrieved from <http://projecthydra.org/design-principles-2/>
- Duraspace (2012). *Hydra objects, content models (cModels) and disseminators*. Retrieved from: <https://wiki.duraspace.org/display/hydra/Hydra+objects%2C+content+models+%28cModels%29+and+disseminators#Hydraobjects,contentmodels%28cModels%29anddisseminators-Don%27tcallitacontentmodel>
- European Parliament, Council of the European Union. (2009). *Directive 2009/136/EC of the European Parliament and of the Council of 25 November 2009 amending Directive 2002/22/EC on universal service and users' rights relating to electronic communications networks and services, Directive 2002/58/EC concerning the processing of personal data and the protection of privacy in the electronic communications sector and Regulation (EC) No 2006/2004 on cooperation between national authorities responsible for the enforcement of consumer protection laws. Official Journal of the European Union, L 337/11*. Retrieved from <http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32009L0136&from=EN>
- Google (2013). *Our Mobile Planet: Ireland, Understanding the Mobile Consumer*. Retrieved from <http://services.google.com/fh/files/misc/omp-2013-ie-en.pdf>
- Government of Ireland. Copyright and Related Rights Act (2000). Retrieved from <http://www.irishstatutebook.ie/2000/en/act/pub/0028/>
- Government of Ireland. Freedom of Information Act (1997). Retrieved from <http://www.irishstatutebook.ie/1997/en/act/pub/0013/>

- Harvard University. (2009). *File Information Tool Set (FITS)*. Retrieved from: <http://projects.iq.harvard.edu/fits/home>
- Hearst, M. A. (2009). *Search User Interfaces* (1st ed.). New York, USA: Cambridge University Press.
- Higher Education Authority. (2013). *National Principles for Open Access Policy Statement*. Retrieved from http://www.hea.ie/sites/default/files/national_principles_on_open_access_policy_statement_final_23_oct_2012_v1_3_0.pdf
- Johnson, J. (2010). *The 960 Grid System Made Easy*. Six Revisions. Retrieved from http://sixrevisions.com/web_design/the-960-grid-system-made-easy/ ACM.
- Johnson, R. (2011). *Atrium EAD Content Modelling*. Retrieved from <https://wiki.duraspace.org/display/hydra/Atrium+EAD+Content+Modelling>
- Kitchin, R., Collins, S., & Frost, D. (in press). *Funding models for open access digital repositories*. Online Information Review.
- Lagoze, C., Payette, S., Shin, E. and Wilper, C. (2005). *Fedora: An Architecture for Complex Objects and their Relationships*. Forthcoming in Journal of Digital Libraries, Special Issue on Complex Objects, Springer
- Mell, P. M., & Grance, T. (2011). *The NIST Definition of Cloud Computing*. Retrieved from http://www.nist.gov/manuscript-publication-search.cfm?pub_id=909616
- O'Carroll, A., Collins, S., Gallagher, D., Tang, J., & Webb, S. (2013) *Caring for Digital Content, Mapping International Approaches*. Maynooth: NUI Maynooth; Dublin: Trinity College Dublin; Dublin: Royal Irish Academy. Retrieved from <http://dri.ie/caring-for-digital-content-2013.pdf>
- O'Carroll, A. & Webb, S. (2012). *Digital archiving in Ireland: national survey of the humanities and social sciences*. National University of Ireland Maynooth. Retrieved from <http://dri.ie/digital-archiving-in-ireland-2012.pdf>
- Online Computer Library Center & Center for Research Libraries. (2007). *Trustworthy Repositories Audit & Certification*. Retrieved from http://www.crl.edu/sites/default/files/attachments/pages/trac_0.pdf
- Project Hydra. (2009). *ActiveFedora*. Retrieved from https://github.com/projecthydra/active_fedora
- Project Hydra. (2010a). *Opinionated Metadata*. Retrieved from <https://github.com/projecthydra/om>
- Project Hydra. (2010b). *Solrizer*. Retrieved from <http://rubygems.org/gems/solrizer>
- Project Hydra. (2012). *ActiveFedora Relationships*. Retrieved date from: https://github.com/projecthydra/active_fedora/wiki/Relationships

- Research Libraries Group and OCLC. (2002). *Trusted Digital Repositories: Attributes and Responsibilities*. USA: Research Libraries Group. Retrieved from <http://www.oclc.org/content/dam/research/activities/trustedrep/repositories.pdf>
- Saracevic, T. (2000). *Digital Library Evaluation: Toward an Evolution of Concepts*. *Library Trends*, 49(2), 350–69.
- Wieggers, K.E. (2003). *Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle*. Microsoft Press.
- Wieggers, K.E. (2006). *More About Software Requirements: Thorny Issues and Practical Advice*. Microsoft Press.
- W3Techs. (2014). *Usage statistics and market share of JQuery for websites*. W3Techs.com. Retrieved from <http://w3techs.com/technologies/details/js-jquery/all/all>
- Wynne, M. & Hellesøy, A. (2012). *The cucumber book: behaviour-driven development for testers and developers*. (Carter, J., Ed.). The Pragmatic Bookshelf.
- Zick, G. (2009). *Digital Collections: History and Perspectives*. *Journal Of Library Administration*, 49(7), 687.

Appendix 1: Cucumber Feature Example

Scenario: Bulk Ingest of a directory 10 of assets and metadata.xml files

Given I am logged in as "user1"
And there is an existing collection
And I have depositor permissions for the collection
And there is a valid "bulk ingest" location
And the "Digital Assets" are valid
And the metadata are valid
When I select the collection
And I specify the "bulk ingest" location
And I run the "bulk ingest" command-line tool
Then the digital objects should be created in the collection
And I should get a "report"
And the report should contain a list of "PIDs"
And the report should contain a list of "URLs"
And the report should contain a list of "checksums"

CollectionsController

create
destroy
edit
new
publish
review
update

_layout
create_from_form
create_from_xml
create_reader_group
delete_collection
publish_collection
reader_group_name
valid_permissions?

MetadataController

show
update

_layout

ObjectsController

citation
create
edit
index
new
related
show
status
update

_layout

CatalogController

enforce_search_for_show_permissions
exclude_unwanted_models
rows_per_page
solr_search_params_logic
solr_search_params_logic=
solr_search_params_logic?

_layout

ObjectHistoryController

show

_layout

UserReportController

index

_layout

DatastreamVersionController

show

_layout

ApplicationController

_logging_in_user_callbacks
_logging_in_user_callbacks=
_logging_in_user_callbacks?
after_sign_out_path_for
duplicates?
retrieve_object
retrieve_object!
set_access_permissions
set_cookie
set_locale
solr_access_filters_logic
solr_access_filters_logic=
solr_access_filters_logic?
supported_licences

_layout
authenticate_user_from_token!
duplicates

TimelineController

get

_layout
cover_image
create_timeline_data
default_image
get_cover_image
get_query
parse_dcmi
parse_dcmi?
search_image
surrogate_url

SessionController

create

_layout

InstitutesController

associate
associate_depositing
create
new
show

_layout

Appendix 2: Controllers Class Diagram

