and

CLIF Project

CLIF technical design

Chris Awre, Richard Green, Andrew Thompson, Simon Waddington

July 2010

The CLIF Project

Project Director: Chris Awre (c.awre@hull.ac.uk)
Project Manager: Richard Green (r.green@hull.ac.uk)
Project Site Manager for King's College: Mark Hedges (mark.hedges@kcl.ac.uk)

The CLIF Project is being undertaken by the Information Systems Group at the University of Hull and the Centre for e-Research (CeRch) at King's College London. It is funded by the JISC Information Environment Programme 'Repositories Enhancement' strand.

Table of Contents

1. Introduction

The CLIF project recognises that

> "At the heart of meeting institutional needs for managing digital content is the need to understand the different activities that the content goes through, from planning and creation through to disposal or preservation. Digital content is created using a variety of authoring tools. Once created the content is often stored somewhere different, made accessible in possibly more than one way, altered as required, and then moved for deletion or preservation at an appropriate point. Different systems can be involved at different points: one of these may be a repository. To embed repositories in the content lifecycle, and prevent them becoming yet another content silo within the institution, they thus need to be integrated with other systems that support other parts of this lifecycle. In this way the content can be moved between systems as required, minimising the constraints of any one system."
>
> *CLIF Project Plan, April 2009*

The project thus set out to investigate how flexible support of such a lifecycle might be enabled in an institution that uses a Fedora-based repository[1] and manages documents using Microsoft Office SharePoint Server[2] (henceforth just 'SharePoint') and/or the Sakai academic collaboration platform.[3]

This document brings together three separate reports that were outlined in the CLIF Project Plan:[4]

- D3 A technical review document
- D4 An ESB review document, and
- D5 Documentation of the proposed CLIF architecture

With the benefit of hindsight it is clear that the three reports are intimately related and it now seems more appropriate to present them as a single text where appropriate cross-referencing can be achieved more easily.

It is hoped that this report will allow technically-aware readers to understand the underlying design decisions taken for CLIF and to reflect on these against the background of their own institution, and thus to consider how an approach similar to that taken by the Project might benefit their own colleagues. Overall, it is hoped that the CLIF project will

> "bring about two key outcomes. The first will be a fuller understanding, shared with the community, of the lifecycles that digital, and especially born-digital, materials undertake. The project will then show how Fedora, Sakai and/or MOSS technologies can be brought together to provide a coherent framework in which to provide integrated management of these objects throughout their lifecycles."

[1] Fedora Commons website: http://fedora-commons.org
[2] Microsoft SharePoint site: http://sharepoint2007.microsoft.com
[3] Sakai Project See: http://sakaiproject.org/
[4] CLIF Project Plan at https://edocs.hull.ac.uk/muradora/objectView.action?parentId=hull:1647&type=1&pid=hull:1808

The choice of software, Fedora, Sakai and SharePoint, reflects the interests of the CLIF partners and advisers. It is intended that the CLIF software outputs will eventually be made available to the community under an appropriate Open Source licence and that it will be possible to deploy code to link just one pair of systems (Fedora and SharePoint or Fedora and Sakai); potential beneficiaries from CLIF would not need an interest in both SharePoint *and* Sakai.

Finally, readers should note that this document represents the team's thinking at a particular point in time and it may well be that we shall change aspects of the CLIF design as the project progresses and as we learn more about the complex interactions that are involved in implementing this work.

2. Aims and objectives

The CLIF Project aims to address the following challenges:

2.1 By linking the repository into other content creation and management environments it will be taken upstream in the user's workflow. Where the repository is best positioned within the content lifecycle requires investigation: it may be relevant at the end of the creation stage to move the content into a repository for access and/or preservation; or it may be appropriate to move content into the repository as a staging area for subsequent processing.

2.2 The aim of integrating the repository at the appropriate part of the content lifecycle is to ensure that, when user activity crosses system boundaries, users do not feel constrained in what they wish or need to do; rather, the systems in question between them support these wishes and needs. For example, moving content used for teaching in a VLE into a repository, maybe as part of building a portfolio, supports content re-use and the potential for long-term access.

2.3 CLIF is starting from a point of agnosticism about the direction content will flow between the repository and other systems (the lifecycle may require movement in both directions). Nevertheless, by facilitating the links between systems it is intended to support preservation by allowing the content to be moved to a system that has preservation capability.

2.4 The development of preservation policies for the repositories as part of the project will guide the technical work proposed. Whilst looking at specific policies for preservation, the potential of incorporating the principles involved into wider institutional policies supporting research, teaching and administration will also be explored, to link the management of the content to the purpose for which it is being managed.

All four of these challenges have had an influence on the work described here.

3. Technical overview

This section gives an overview of Fedora, SharePoint and Sakai and outlines what the CLIF Project will initially try and add by providing a level of integration. The section relating to Fedora is necessarily longer than the other two in order to provide the reader with a 'Fedora primer' sufficient to the understanding of some of the technical discussions that follow in later sections.

3.1 Fedora

3.1.1 About Fedora

To quote from Fedora's own documentation:[5] "Fedora is an acronym for the *Flexible Extensible Digital Object Repository Architecture*. The Fedora Repository is very flexible; it is capable of serving as a digital content repository for a wide variety of uses. Among these are digital asset management, institutional repositories, digital archives, content management systems, scholarly publishing enterprises, and digital libraries. The Fedora Repository is able to store any sort of digital content item such as documents, videos, data sets, computer files, images plus it can store information (often called metadata) about the content items in any format. In addition, the relationships between content items can be stored - which is often as important as the content items themselves."

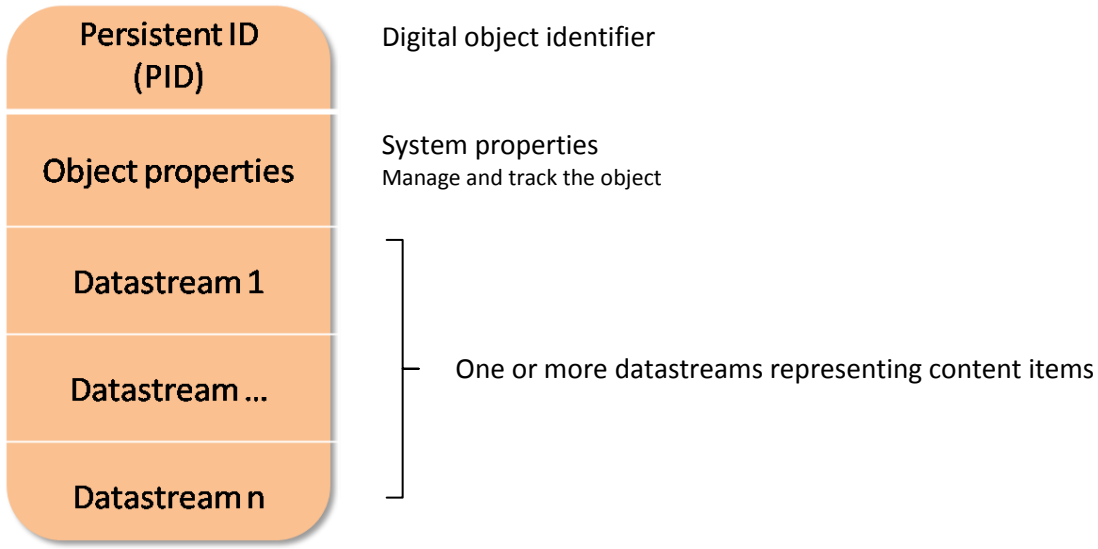
In the context of the CLIF project, Fedora is important as the basis for a number of repositories, institutional or otherwise, in the UK Higher Education sector. Its flexible nature means that digital objects within a Fedora repository, the containers for each piece of digital content, can be structured in many alternative ways and we anticipate that this will bring its own set of challenges.

Fedora has no out-of-the-box general user interface and thus must (and does) provide application programming interfaces (APIs) which allow all the necessary interactions with the content of a repository. However, the very flexibility of the Fedora system means that there may be multiple ways of using these to achieve similar ends. The APIs are provided through Web Service Interfaces (both REST and SOAP). In addition to the APIs, Fedora provides a Simple Java Messaging Service (JMS) as middleware and this may well prove to be useful in CLIF's work.

[5] Fedora Tutorial 1 at: http://fedora-commons.org/confluence/download/attachments/4718930/tutorial1.pdf?version=1&modificationDate=1218459761506

3.1.2 Fedora objects

In order better to understand the operation of Fedora we should first address the principles underlying the structure of Fedora digital objects, the Fedora Digital Object Model. Provided here is a simple (perhaps even simplistic) summary of what can be a complex topic.



Each object has a persistent, unique identifier. It then has a set of system-defined descriptive properties that are necessary to manage and track the object in its repository. Finally there are a number of datastreams which pertain to the digital content, the payload, of the object: some of these may represent the content itself, others may contain metadata about the content, there may be datastreams describing relationships between this object and others and there will be a system-provided 'audit' datastream wholly managed by the system to record changes made to the object.

Any Fedora object contains a Dublin Core (DC) metadata datastream; if the object's creator does not provide one Fedora itself will generate it. At the least this contains a title for the object and a formal record of its PID. This information is indexed and provides the basis for Fedora's basic administrative search mechanism. In addition, a 'simple' Fedora object will contain only one content datastream, perhaps representing a text document or a PowerPoint presentation. A 'compound' Fedora object may contain several content datastreams; best practice would suggest that these should all be the same content but expressed in different ways, perhaps the same image provided in a range of resolutions or the same text document in a variety of formats. A 'complex', sometimes called 'atomistic', Fedora object actually comprises a set of simple and/or compound objects. The 'parent', or 'aggregation', object generally contains information about the remaining related 'child' objects; it has no digital content itself but describes and identifies the children. Thus a thesis consisting of a PDF text and three video clips might be represented by a complex object – a parent which describes the thesis, the clips and their relationships, and four simple objects each providing access to one of the files involved. The files of digital content can be managed by Fedora itself but it is probably more common in practice that they are held in an http(s) accessible file store and the Fedora object contains URLs that allow the repository to retrieve them. In theory, a content datastream can represent any form of digital content; put another way, Fedora is content agnostic.

The default configuration of Fedora allows for versioning. Each time a datastream is changed a new version of it is created, thus allowing a user to revert to or otherwise use an earlier version should that be desirable. If Fedora itself manages the digital content, that too is versioned; if not, the local system must be designed to keep versioned copies of the content in the external store.

The information about the object and its datastreams is represented using XML. Thus, the file on a Fedora server which represents a Fedora object contains XML, specifically Fedora Object XML (FOXML) which has a formally defined schema.

3.1.3 Content models

Since the release of version 3 of Fedora, the software has supported the concept of a Content Model Architecture (CMA) that is used to describe the common structures of objects in a particular repository. Use of the CMA encourages (and potentially ensures) internal standards within a repository and allows the use of one particular feature of Fedora that may prove useful in CLIF's development work, 'disseminators'.

3.1.4 Disseminators

Disseminators can provide an abstraction layer between a Fedora object and the retrieval process. Instead of retrieving the contents of an object's datastream directly, the process can be mediated through a disseminator which can transform it on the fly. Thus, for instance, a Word document stored in a Fedora object could be called using a disseminator method that converts it and delivers it to the user's browser as a PDF. Although the idea is speculation at this stage in CLIF's development, there may be a case for using this technology when transferring content between Fedora and either MOSS or Sakai. Disseminator functions are not available in the reverse direction: it is not possible to have a disseminator convert the format of content being deposited in the repository. However, there are deposit mediation tools available to the Fedora community and particular mention should be made of one of them, SWORD.

3.1.5 SWORD

The Simple Web-service Offering Repository Deposit (SWORD) tool is available as part of the Fedora Framework Services. This JISC-funded software offers a common way to deposit objects into any of the three repository types common in the UK: Fedora, DSpace and EPrints. SWORD's deposit process is capable of producing simple or compound Fedora objects. It appears from the web documentation provided[6] that a very limited set of file types (MIME types) is accepted directly but that other types may be accepted if offered as part of a METS package. The potential applicability of SWORD to CLIF's work has been considered (see the introduction to Section 4).

[6] See: http://www.swordapp.org/sword/demonstrators

3.1.6 Fedora standards

As we have noted above, Fedora is immensely flexible and so it would be appropriate to try and adopt a 'standardised' approach to it for the CLIF Project in order that any project outcomes may be re-usable in a straightforward fashion. The Hydra Project[7] provides just such a set of standardised approaches; in particular it provides a consistency of approach to content models and associated disseminators. The CLIF Project will adopt the Hydra principles for the construction of content models, digital objects, disseminators and objects. Whilst Hydra provides a basic set of content models, CLIF will need to supplement these with some of its own. The additional content models and disseminators that may be required will be designed to be consistent with Hydra's approach. The Hydra approach itself, tries to make a minimal number of assumptions about a repository and thus we hope that CLIF-generated Fedora objects will be widely useful.

3.2 SharePoint

3.2.1 About SharePoint

According to the Microsoft SharePoint website:[8]

"Microsoft SharePoint … makes it easier for people to work together. Using SharePoint, your people can set up Web sites to share information with others, manage documents from start to finish, and publish reports to help everyone make better decisions." Put very simplistically, amongst other functionality, SharePoint adds web-based collaborative facilities to the Microsoft Office family of software.

The Northumbria University *Investigation into the use of Microsoft SharePoint in Higher Education Institutions: Final Report*[9] summarised the use of SharePoint in UK HEIs and identified a range of drivers across the implementations that they studied:

a) to improve document management: SharePoint document libraries can be deployed with version control, check in and check out, and metadata capture
b) to support collaboration with external partners: if an organisation has an external connector licence they can add external people to their Active Directory, or equivalent identity management system, and allow site owners to provide them with access to particular SharePoint sites
c) to improve cross-school/departmental working: site owners can allow access to sites to colleagues anywhere within the institution without having to go through an IT administrator
d) to enable staff and students to find colleagues with similar interests: in SharePoint 2007 My Sites can make a profile of an individual available for the rest of the organisation to view or search on, though the take up of My Sites amongst staff and students has been relatively low

[7] See: http://fedora-commons.org/confluence/display/hydra
[8] See: http://sharepoint.microsoft.com/en-us/Pages/default.aspx
[9] See pages i-ii of the report at: http://www.northumbria.ac.uk/sd/academic/ceis/re/isrc/themes/rmarea/eduservsp/

e) to improve an intranet or external website: SharePoint is more frequently used for intranets rather than the management of external websites. Customisation is required if an HEI wants a website in SharePoint 2007 to meet Web standards, including accessibility standards. Customisation is also required if an organisation wants to apply its branding to SharePoint sites
f) to target information (typically through an intranet) to particular audiences
g) to improve and automate cross-institution processes: though the institutions in the study have not found Infopath electronic forms easy to get set up and working
h) to provide a personalised portal where staff and students can log-in to one place and access all the different systems of the HEI: SharePoint has a single sign on facility, although the HEI needs to write some custom code to get it to work
i) to bring together and manage data from different information systems around the organisation: for example, in order to pull data from student databases and finance databases and display and manipulate it in the SharePoint environment.

From these observations we can usefully extract some key points to inform the CLIF Project.

It is clear that a significant number of HEIs in the UK are using instances of SharePoint for a wide range of purposes. Significant amongst these, from the point of view of CLIF, are those instances where SharePoint is used in the (possibly collaborative) production of materials that will eventually have a wider audience than those individuals who have contributed to their authoring. Such materials might include, amongst others, policy documents, agendas and minutes from meetings, learning materials, reports, and so on. One can easily envisage a situation in which SharePoint is used as the authoring environment, taking advantage of its facilities for collaboration, versioning and the like, but where the final document is made available to a much wider audience by placing it in an institutional repository. Equally, one might envisage in the case, say, of a policy document that the current version in a repository might be used as the starting point for a subsequent revision. Thus one has the need to be able to transfer a document from SharePoint to the repository and later, *vice versa*. (We have talked here in terms of document, but the same argument can be made for any form of content that might be originated within the Microsoft Office family of programs - and which was therefore suited to manipulation in SharePoint - including numeric data and databases.)

This is quite a useful example for understanding CLIF's approach to the content lifecycle. During authoring or re-authoring, SharePoint would be an appropriate environment in which to work. For exposing the content to a wide audience an institutional repository would probably be considered appropriate. Going beyond that, a repository is probably far more suited to long-term preservation of the content should that be appropriate. The CLIF Project is therefore interested in providing software that will allow the easy transfer of materials between SharePoint and a Fedora-based repository.

3.2.2 Site documents

The 'Shared Documents' area of a SharePoint site provides a drop-down menu for each document consisting of

- view properties
- edit properties
- delete

- version history
- connect to client, and
- alert me

In order to perform the simplest possible interaction with Fedora, that of copying a SharePoint shared document and from it creating a single Fedora repository object, CLIF will seek to add into this list

- copy to repository

and provide the appropriate functionality. Reverse functionality, to fetch the content of a Fedora object into a SharePoint site, will also be developed.

3.2.3 SharePoint workflows

SharePoint provides the facility for users to define workflows to assist in the production of content. Consider this generalised and simplistic example:

A team of three people work together to produce an undergraduate programme of study: the course leader, the Head of Department and an external examiner. A workflow is set up in which the course leader first creates a course specification. Once this document is complete, the workflow tools allow him to send a request to the Head of Department to review and approve it. The Head of Department will receive an automated e-mail requesting that he log into the appropriate SharePoint site and do this; on completion of the work the workflow tools allow the sending of a response to the course leader approving or rejecting the documents and providing any appropriate comments. This cycle can be repeated until approval is received. Once the Head of Department has given this approval then, and only then, does the workflow allow the course leader to ask the external examiner, via another automated e-mail, to log in and likewise accept or reject the documents.

In a CLIF-enabled world, the workflow would ultimately allow a final stage in which, with all approvals in place, a copy of the programme of study could be made available in the institutional repository. This would involve taking a copy of the approved document, using it as the basis for creating a digital object in the Fedora repository and additionally populating that object with appropriate metadata. The metadata would potentially cover a range of areas: metadata about the content of the document; technical metadata about the document file; metadata about any security considerations; metadata relating the document to other, similar, documents in the repository; metadata describing the history of the content within SharePoint; and so on. This last, metadata describing the object history, will be an important element in providing provenance information.

Thus the CLIF Project hopes to provide additional software which allows a SharePoint user to add this repository submission stage into a normal SharePoint workflow and to do so in such a way that the tool appears as an integral part of the SharePoint web interface.

Conversely, we hope to provide the reverse functionality; the ability to take a copy of digital content in the Fedora repository into the SharePoint environment in order to contribute to a workflow in some way.

3.3 Sakai

3.3.1 About Sakai

To quote from the Sakai Foundation's web site:

"Leading educational institutions throughout the world choose Sakai to enable powerful teaching and learning and research collaboration. Depending on where you are in the world, Sakai might be called a Course Management System (CMS), a Virtual Learning Environment (VLE) or Learning Management System (LMS). While Sakai is typically used for teaching and learning (similar to products like Blackboard and Moodle) we call it a Collaboration and Learning Environment (CLE) because it embraces uses beyond the classroom." Omitted from the list is its use in a number of UK institutions as a Virtual Research Environment (VRE).

Within the Sakai environment is provided a 'resources' area where the members of a collaboration, of whatever sort – learning, research or other – can share a range of useful materials. In an HE environment it seems reasonable to assume that some such resources might exist within an institutional repository. One solution, of course, is to copy a resource from the repository into a particular resources area. A more general solution might be to make the repository, or part of it, appear in the Sakai resources tool alongside resources specific to that Sakai collaboration. Alongside this one can see merit in being able to transfer a Sakai resource to the repository in order to be more generally available, or to transfer a copy of a resource from the repository to Sakai – perhaps so that it can be used in a customised form.

Thus, the aim of the CLIF Project is to enable such functionalities within Sakai in an integrated fashion.

3.3.2 CTREP

A previous JISC funded project, CTREP,[10] sought as part of its work to integrate the Sakai resource tool with a Fedora repository. It seemed appropriate that CLIF's initial work with Sakai should seek to evaluate this legacy codebase and to determine whether it offered a starting point for our own work or whether a different approach should be taken.

3.3.3 The Sakai resources tool

The resources tool offered by Sakai allows an appropriately authorised user to create a tree structure of resources, providing functionality via a drop-down menu to

- upload files (to the current folder)
- create sub-folders (within the current folder)
- add web links
- add a citation list
- create an HTML page, or
- create a text document

At the individual resource level functionality exits to

- copy

[10] See: https://camtools.cam.ac.uk/access/wiki/site/jisc-ctrep/home.html

- edit details
- upload new version
- move
- remove, or
- duplicate

The initial plan is that CLIF should add functionality to the first group to

- add repository as folder (within the current folder)
- remove repository from folder

and add to the second group as applied to a repository object

- download content (to containing folder)

and as applied to a non-repository object

- upload to repository

Further refinement of these functions may take place if and when the initial functionality is established.

4. Integration investigation

A great deal of discussion has taken place around the twin challenges of how to integrate Fedora with Sakai and SharePoint. In the end we have followed the 'KISS principle' (Keep it simple, Stupid!) and tried to develop software that will be easily deployed by other potential users and which has few necessary dependencies. Two discarded approaches should perhaps be mentioned in this introduction.

We noted at quite an early point in our deliberations that Fedora apparently has the ability to import METS packages and wondered whether we could use METS as a transfer format between our pieces of software. If so, our logic went, then potentially any software that could import or export METS packages should be easily adaptable to the CLIF approach. Further research revealed that Fedora uses a rather heavily customised form of METS and that "arbitrary METS files will not necessarily be meaningful" [to Fedora].[11] Whilst the possibility of working with XSLT transforms might have allowed us some success with the METS idea, we felt that we were straying too far from the simplicity that we had hoped for.

The JISC-funded SWORD (Simple Web-service offering Repository Deposit) Project has developed a "lightweight protocol for depositing content from one location to another"[12] which can be used to deposit content into Fedora. In its relatively short lifetime SWORD has attracted many adherents.

[11] Thornton Staples, Director of the Fedora Project; *in litt* 2010-04-09
[12] See the SWORD website at: http://www.swordapp.org/

However we rejected its use as a deposit mechanism in CLIF for two reasons. Firstly, it is what it says "on the tin". It is a deposit mechanism and, at the time of writing, has no mirror functionality to fetch content from a repository – a necessary component of CLIF. In addition, when an object is deposited using SWORD the returns from the software are not currently very helpful in terms of feeding back useful information for the user (be that user machine or human); work is apparently in hand to improve this aspect of the tool.

At the time of writing, therefore, we are producing a system that creates a FOXML file as the transfer vehicle. In the near future we may evaluate the possibility of using OSIDs (Open Service Interface Definitions) as a transfer mechanism between Sakai and Fedora.

It is not the intention of CLIF to create Fedora objects from within SharePoint or Sakai "just" so that they can go into a Fedora repository. CLIF addresses the idea of a content lifecycle and so, in creating Fedora objects, we have tried to look beyond immediate and obvious needs to consider longer-term functionality. To that end, objects created in Fedora will contain metadata datastreams that record, for instance, contextual information about the original SharePoint or Sakai object, technical information derived from JHOVE,[13] preservation information derived from DROID,[14] and other such information that might prove useful at other points in the life of the material.

Finally we should note that our proposed development does not involve the use of an Enterprise Service Bus (ESB), a possibility raised in the initial Project Proposal. The ESB consideration is dealt with more fully at section 5.

4.1 SharePoint-Fedora integration

4.1.1 Overview

The two basic requirements for the SharePoint-Fedora integration are to deposit content items and associated metadata from SharePoint to Fedora, and to search, browse and retrieve items stored in Fedora from the SharePoint user interface.

A first mechanism for deposit to Fedora will be achieved by manual interaction by the user. From a pull-down menu associated with a document in a 'Document Workspace', the user will be able manually to deposit items into the repository or a repository approval queue. (It may not be desirable that a SharePoint user have the ability to create and expose an object to public view without a mediation and/or quality control step, hence the need for an approval queue.)

Secondly, in order to achieve the objectives of CLIF in moving the repository upstream in the content lifecycle, a study was carried out with a view to enabling the Fedora deposit to be attached to key steps in the creation and approval of content. The natural way to implement this within SharePoint is to integrate the deposit with a SharePoint feature, which include both workflows and web parts.

SharePoint workflows provide a mechanism to perform a structured sequence of tasks, which are typically performed by a predefined set of users. For example, workflows provide a convenient way to implement document approval processes.

[13] JHOVE – the JSTOR/Harvard Object Validation Environment: See: http://hul.harvard.edu/jhove/
[14] See: http://www.nationalarchives.gov.uk/aboutapps/pronom/ for information about DROID and PRONOM

Web parts are the basic building block for web pages in SharePoint. A web part is a server control which is added to a web part zone on a web page at run time. The controls enable end users to modify the content, appearance, and behaviour of web pages directly from a browser. Web parts provide a convenient and flexible method to integrate data from multiple applications into a single web page.

4.1.2 Scenarios

In order to investigate the technical issues in the deposit of content items from a SharePoint to Fedora, four scenarios were defined and implemented. These scenarios provide technical challenges representative of the main use cases being considered in the CLIF project.

1. Manual deposit to Fedora.

- A user creates a document in SharePoint within a Document Workspace.
- The user selects Copy to Fedora from the options in the pull-down menu associated to the content item.
- The item is deposited to Fedora.

2. Creation and approval of examination papers.

- A lecturer creates draft documents (exam paper, solutions) in SharePoint.
- The lecturer starts the workflow and completes the required submission form, entering privacy level, subject details, reviewer and approver details, exam date.
- The reviewer receives an email notifying them of a task to review the document. The reviewer enters their comments and approves or rejects the documents.
- If the document has been approved, the final approver receives an email notification of a review task. The final approver can either approve or reject the document. The final approver has the option of enabling deposit to the repository. If the document is approved, the Fedora deposit is initiated, and the documents and any metadata are transferred to the Fedora deposit queue.
- If at either of the two review stages, the document is rejected, the lecturer receives an email notifying them of a task to modify and resubmit the document.

3. Creation and approval of a policy document.

- A committee secretary requests a new SharePoint site to host policy documents.
- The SharePoint administrator creates a workflow to enable deposit from the SharePoint site to Fedora.
- The secretary runs the workflow on a draft document, adding privacy settings, reviewers and approver names and retention period to the submission form.
- As in scenario 1, iterative review tasks are completed by the reviewers and approvers.

- A PDF version is published to a public area in the repository. Audit trail, approval forms, and change logs for the document are archived within the Fedora object, but secured from public view.

4. Environmental modelling calculation.

- A researcher creates an Excel spreadsheet and stores it in a trusted location in a SharePoint document site.
- The researcher opens the 'Calculation' web page and selects a data set to import into the spreadsheet using the web part controls.
- The user runs the calculations and views the outputs.
- The user elects to store the data in the repository. The user can enter additional information in a form. Once this form is submitted, the input and output datasets, the spreadsheet, and additional metadata are deposited to Fedora.

4.1.3 SharePoint-Fedora deposit

Figure 4.1 illustrates the steps required to implement scenario 1. A "Copy to Fedora" option was implemented on the menu associated with each document in a 'Document Workspace'. A Fedora object was then created containing basic metadata (Dublin Core), ingested to the repository via an object queue and there identified for mediation and/or quality assurance (QA).

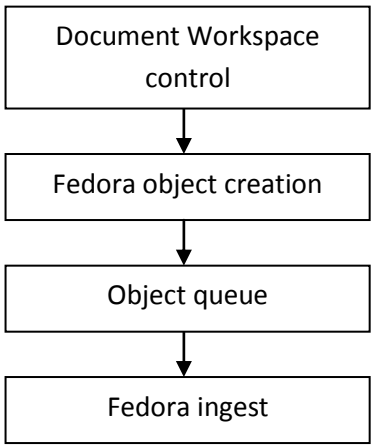


Figure 4.1 SharePoint-Fedora deposit from a user control in a Document Workspace

The main steps required to implement scenarios 2 and 3 are illustrated in Figure 4.2. The most suitable SharePoint feature was the State Based Approval Workflow provided in the SharePoint SDK examples. This provides a two step approval process which is run on a selected document in a document workspace.

The workflow allows an administrator or document author to select a reviewer and approved from the list of SharePoint users. When the workflow is run, the reviewer receives an email notification of a task to complete the review of the document. The reviewer has the option to enter comments and either approve or reject the document. If the document is approved, the approver then receives an email to notify them of the approval task, which is completed in a similar fashion to the reviewer. The approver can select whether the document is deposited to Fedora. If the document is approved,

the Fedora deposit process is executed and the workflow is completed. If either the reviewer or the approver rejects the document, a task is created for the document author to revise the document. Once this is completed, the approval process can be repeated.

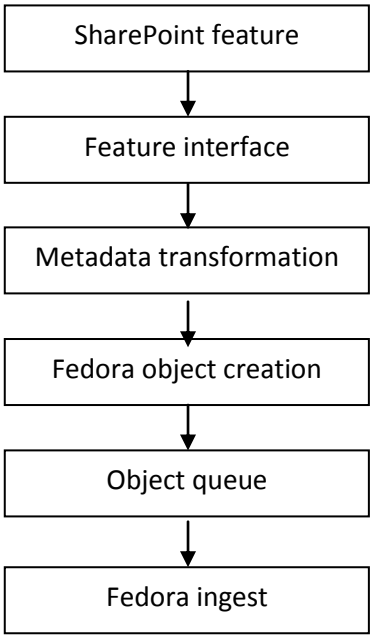


Figure 4.2 SharePoint-Fedora deposit from a feature

An API was defined to allow capture of the document within the workflow together with metadata associated with the workflow stored within SharePoint. This API is referred to as the feature interface. It includes capture of audit and workflow history information that is automatically logged within SharePoint as well as the user identities and comments entered by the reviewer and approver. SharePoint also provides the facility to capture multiple versions of the submitted document through the version control functionality.

4.1.4 Metadata transformation

Metadata transformation involves mapping of SharePoint metadata onto standard formats used in digital preservation such as Dublin Core and MODS. Simple conversions that are unlikely to need modification were carried out in software. An alternative for more complex transformations that may require change is to use an XSLT.

At an early stage it was considered whether to implement the metadata transformation and object creation within Windows SharePoint Services (WSS) or to use Java middleware. There appear to be few differences between the approaches apart from the issue of possible code reuse from other related projects. In this case, it was decided to implement this step in C# within WSS.

A further step that is required to assist in long-term preservation is document format conversion. For example, highly proprietary document formats such as Word may be converted to PDF format for deposit in the repository. Document conversion can be carried out either within Windows SharePoint Services using for example the open source Doc2Pdf software that integrates directly

into SharePoint[15], by calling an external converter as a service or by providing additional Java middleware to perform the conversion prior to ingest into Fedora. Conversion of documents within WSS would require load balancing and scheduling within the SharePoint server installation, for instance by providing a dedicated server for document transformations.

4.1.5 Fedora object creation

Fedora object creation is the process of creating the required XML files accepted by the Fedora API-M services. The chosen ingest format was FOXML. The object creation includes the creation of the Fedora mandated DC datastream as well as the RELS-EXT datastream used by many repositories to define structures. Additionally, creation of audit trails and other application-specific datastreams is carried out.

4.1.6 Authorisation and policy management

In order to effectively manage authorisation within Fedora, creation of policy data streams was investigated. Fedora has adopted XACML as a policy language. This allows complex rules to be defined that limit access to raw objects, disseminators and web service APIs. Fedora allows authorisation to be managed at the repository level, via repository wide XACML policy documents, or at the Fedora object level using a policy data stream included in each individual object. Repository wide policies are the simplest approach when there are a relatively small number of policy requirements across the repository. When there are widely differing requirements across document sub-collections, repository wide policies can become complex and increase response times in accessing documents. Each request made to the repository requires validation by the XACML engine prior to execution.

Since some Fedora repositories may be required to archive documents from multiple SharePoint sites with differing access requirements, creation of policy datastreams in each object will be investigated in addition to allowing the 'repository wide' approach. In order to reduce the complexity of maintaining a large number of policies, these policy datastreams will be configured to be externally referenced. In this way, Fedora objects with the same authorisation requirements (for example originating from the same SharePoint site) can share a single expression of the policy. Policies may be subject to change in the period that a document is retained in the repository.

4.1.7 Object queue

In order to decouple the Fedora ingest process from SharePoint, creation of an object queue was investigated. Fedora installations typically make use of the Apache ActiveMQ Java Messaging Service (JMS). The messaging service provides updates about the activities of the repository as they occur. A reference implementation of the Java messaging client is provided by DuraSpace.[16] An alternative JMS that is being considered for use with Fedora is RabbitMQ.[17] However, currently no documentation to support integration with Fedora is provided. This object queue should not be confused with the approval queue mentioned in 4.1.1.

4.1.8 Fedora ingest

Fedora ingest is performed by calling the Fedora administrator web service API-M. For determining the locations of items in the repository, two separate use cases were identified. In the first case,

[15] See: http://docconverter.codeplex.com/
[16] See: http://www.fedora-commons.org/confluence/display/FCR30/Messaging
[17] See: http://www.rabbitmq.com/

Fedora objects are moved manually from an approval queue to a location in the repository. This task is performed by a librarian or curation specialist as part of QA and mediation work. In the second case, the Fedora objects would be routed automatically. The most logical form of automatic routing would be to organise the repository structure to mirror the site and site collection structure of SharePoint.

For automated object ingest, the Fedora PID and RELS-EXT need to be generated prior to ingest into Fedora. The method under investigation is to provide an administrative tool to enable creation of a reference Fedora object for each new site collection or site created. In that way, any object deposited from a given site could be added to the relevant collection in Fedora. Fedora PIDs can be generated automatically by Fedora or generated by an external application prior to ingest. The most transparent method was to derive the Fedora PID from the URL of the corresponding item in SharePoint. This provides the additional advantage of making the PID human readable.

Once a reference object has been created for a given SharePoint site, objects can be added to the corresponding Fedora sub-collection by adding a reference in the RELS-EXT data stream.

4.1.9 Extensions of the basic scenarios

For creating more general SharePoint-Fedora interface, a generic feature interface method will be developed that will allow any workflow or web part to be connected to the Fedora repository using the sequence of steps described in Figure 4.1. In the following work, we will aim to demonstrate how arbitrary workflows and web parts can be integrated with a minimum of coding.

A further generalisation we have investigated is to allow the deposit of multiple documents in a single workflow. For example, the solutions for an exam document could be appended to a workflow running on the corresponding exam paper document, in order that both documents can be approved in a single workflow process.

Scenario 3 illustrates the possibility of depositing multiple Fedora objects from single workflow, for instance to deposit the document into a public and a private areas of the repository. The private area would reference detailed metadata and an editable version of the document, whereas the public area would reference a non-editable PDF version. This can be achieved by executing the metadata transformation and object creation repeatedly with different parameters. Note that the documents themselves are typically stored in a separate file store, which is referenced by the Fedora object using the externally referenced configuration setting.

For the fourth scenario, the main difference to scenarios 2 and 3 is the use of a web part as a front end to access the Fedora Excel web services. In a similar way to scenarios 2 and 3, the feature interface is used to capture data from the web part, in this case an Excel spreadsheet, and deposit the document and metadata into Fedora.

Web parts provide a convenient method for displaying content from other applications within a web page. In a similar way to workflows, it is possible to integrate the Fedora deposit functionality with the web part code. In the case of a web part, the deposit can be initiated from a user control in the web part, such as a button.

4.1.10 Fedora-SharePoint browse and retrieval

In order to implement searching of the Fedora repository, indexing of the content is required. GSearch (the Fedora Generic Search Service) is the most common search service used with Fedora and this uses Lucene, or by extension, Solr.

SharePoint provides an indexing service to enable search of content stored within SharePoint sites. The indexing service can also be extended to search external content. Since the use cases we have defined do not have a specific requirement to perform a federated search across both SharePoint and the Fedora repository, we have chosen to implement the indexing of Fedora independently from SharePoint.

4.2 Sakai-Fedora integration

4.2.1 Overview

As noted in section 3.3.2, the CLIF team was aware of work undertaken by the Cambridge Tetra Repositories Enhancement Project (CTREP) and elected to use evaluation of this project as a starting point for Fedora-Sakai integration rather than re-invent the wheel. Specifically, we elected to investigate the Fedora component developed at the University of Highlands and Islands.

The CTREP Fedora - CHH handler is an extension to the Sakai Resources Tool that provides a snapshot tree view of the hierarchical resources within a Fedora repository. This is achieved through the Content Hosting Handler layer (developed by the CARET project) extension mechanism (a provider bean injected into the Sakai AXIS framework and a mount point file uploaded via the resources tool). Full two way editing is apparently implemented.

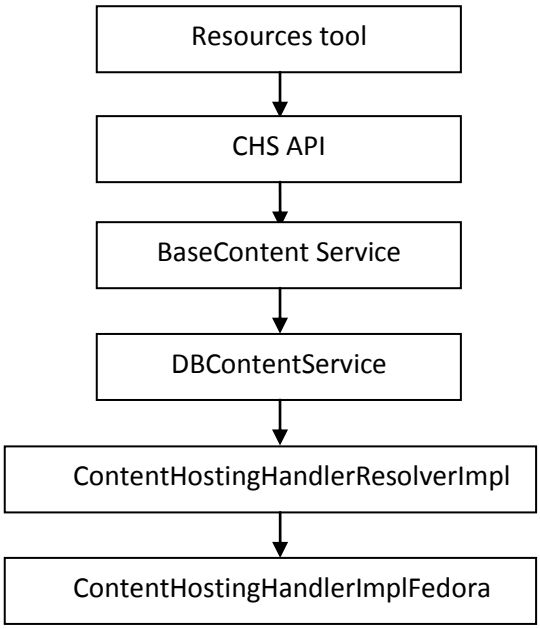


Figure 4.3 Diagram of Sakai Content Hosting Handler model

The structure of the Content Hosting Service is a *BaseContentService*, that implements the API, and then an extension class that specializes the *BaseContentService*

CLIF's work uses Sakai 2.6.1 and Kernel 1.0.12. These were deemed the most suitable stable releases available at the time the CLIF project began.

Because the CTREP code was built against earlier versions of Sakai, some problems were encountered when using the MAVEN build tool (some hand editing of CTREP POM files was required to add some additional jar dependencies and to change the deployment target version).

4.2.2 Interface Description

The CHH Fedora handler communicates with the Fedora system using two SOAP web service APIs already within Fedora; API-A and API-M. The CHH Fedora handler source code makes use of a Guanxi HTTPs transport layer to communicate with Fedora. This requires Java keystore (JKS certificate) security setting up on both client and server and makes it difficult to debug in a development environment as all SOAP messaging traffic is encrypted. After some initial problems setting up the system it was found to work acceptably (albeit very slowly). It was decided to replace this layer with the more basic HTTP protocol not only in order to try and boost performance but to facilitate easier debugging. There is also questionable benefit in securing the connection between Fedora and Sakai as both servers are likely to be on the same security partitioned part of a campus network anyway and HTTPs does require more server processing cycles than HTTP.

4.2.3 Performance Improvements

Quite early on it was apparent that the speed of rendering of resources from the Fedora repository within the Sakai resources tool was very poor (over a minute to render a dozen or so resources within the Fedora repository). The following code re-working has been done so far to alleviate this (mainly in and around *FedoraDigitalRepositoryImpl.java*):

1. Refactor direct calls to web service method as persisted properties (populate multiple properties from one web service method call)
2. No pre-fetch of content datastream for each resource.
3. Caching of resources

4.2.3.1 Refactor direct calls to web service method as persisted properties (populate multiple properties from one web service method call)

Methods *isInCollection*(DigitalItemInfo item) and *isCollection*(DigitalItemInfo item) both made two direct SOAP webservice API calls (approx 3 seconds duration each) and these were used ad-hoc throughout the code wherever these 'properties' of a particular resource were required. The code was re-factored to add two new methods on class *DigitalItemInfo*, namely *isCollection*() and *isInCollection*() and to populate these once on the initial fetch of all top level resources.

4.2.3.2 No pre-fetch of content datastream

The content stream (as well as the metadata streams) of each resource returned from the Fedora resources query would be fetched from Fedora in order to populate the *binaryContent* and *Size* properties on each Sakai *DigitalItemInfo* object. This is very wasteful of server memory (especially since only a few of the potentially large number of resources are ever likely to be

opened to view the actual content). So after some thought, it was decided to adopt a 'just-in-time' read methodology. Instead of pre-fetching the content datastreams, the *binaryContent* buffer would be left empty until actually required. The *contentLength* property was obtained via an alternative API method which only returned basic object metadata. However a bug in Fedora meant that this size (when populated) was always returned as zero for Fedora 'managed' datastreams (issue FCREPO-64). This is scheduled to be resolved in the near future (Fedora 3.4 release). However, an interim fix was made to the Sakai kernel whereby should the CHH-handler return zero length the content is streamed initially to determine its size (which ends up in the HTTP Content-Length header) before streaming again to the user (streaming twice means that a small buffer size can be used saving server memory). This may be a better long-term solution to the problem anyway, because Fedora does not provide for returning the filesize of 'external' content which is the way that many, if not most, Fedora repositories store content.

4.2.3.3 Caching of Resources

A software cache was implemented to store resources fetched during their initial retrieval. Subsequent revisits or refreshes of the Resources Tool Page reads resources from the cache rather than from Fedora via web services. This then makes speed of rendering only a potential issue after the initial fetching of resources. With a reasonable Fedora content collection structure in place it is hoped the number of resources returned will be manageable.

4.2.3.4 Kernel Alterations

Because of the move to a 'just-in-time' read design a change was made to the org.sakaiproject.content.impl .BaseContentService handleAccessResource method to handle resources with a reported 0 size (see previous explanation on this). This modification causes a default buffer size to be allocated, and the content datastream to be read an additional time in order to determine the size of the content (required by the HttpServletResponse.setContentLength method).

4.2.4 Authentication / Authorisation

CHH-Fedora authenticates all web service requests using the user credentials specified in the mountpoint.properties file. In order to restrict which Fedora objects a particular user can access / modify the following approach is likely to be adopted.

At Hull, Sakai user accounts and Fedora users account ids are identical and CAS authentication is used to authorize access. Dealing with access rights on a user's objects then boils down to :

4.2.4.1 Resource object creation / modification Implementation Strategy

Determine / create unique top level root collection per Sakai user with a known namespace:PID (e.g. at hull this would be hull-private:nnnnn where nnnnn is a numeric PID created the very first time a particular user uploads a new object).

In the RELS-EXT datastream for the object, set parent relationship to point at the user's root object above. Set the ownerId of the object to be the user's Sakai account Id. In the DC datastream for the object set the <dc:title> = user's AccountId and the <dc:description> to something searchable in a Fedora search query e.g. 'user root collection'.

4.2.4.2 *Resource listing / display Implementation Strategy*

Using the Fedora Search API issue a query for the user's private root collection PID and any other top level public access collections which have been included in the sakai configuration. The query results will then be displayed as one or more collection folders. A user selecting one of these folders will cause Fedora to isssue a query for all objects (including possible sub-folder objects) with RELS-EXT isMemberOf the selected collection.

4.2.5 Content Formats

The CTREP code at present adopts a fairly generic version of Fedora FOXML 1.0 for its content format. In due course, CLIF will switch this to the current FOXML 1.1 format and implement Hydra content models as standard.

5. Enterprise architecture modelling: Enterprise Service Buses (ESBs)

The CLIF Project Proposal and the subsequent Project Plan promised "Integrations between systems can be carried out using point-to-point techniques according to specific need. Whilst a loosely coupled approach to point-to-point can enable wider adoption of a solution, such solutions can also be limited by the systems themselves as they change over time. Enterprise Service Buses are an approach to abstract out the ways that systems can communicate with each other, protecting integrations against software changes. This final piece of review work will specifically examine available options for using an ESB-approach to inform subsequent technical development."

This is an appropriate juncture at which to consider that review.

To take a phrase from the book *The Definitive Guide to SOA: Oracle Service Bus*[18], "Enterprise service buses (ESBs) are all the rage in modern software development." And whilst they are no more the silver bullet to solve all problems than were XML or web services (capitalised or not) they clearly help to address a particular set of problems. The intent for the CLIF project of reviewing ESBs was to ascertain whether the problems we faced in integrating Fedora with SharePoint and Sakai are a set that could be aided by taking an ESB approach.

The problem space that CLIF has set out to investigate is built on a generic principle about the ability to manage the digital content lifecycle across different systems, so that each can be used to its best advantage in supporting different stages of the lifecycle. The focused implementation of that problem space for CLIF is how content can be moved between SharePoint and Sakai, as two user-facing systems used to manage digital content for different purposes, and Fedora, as a repository system that can support digital content management. The degree to which an ESB is of value needs to be considered at these two levels of scope, and may have different outcomes.

As mentioned in the CLIF Project Proposal, specific needs for integration can be carried out via a point-to-point approach. When there are only a small number of systems involved in the integration, and for CLIF we are limiting ourselves to three, this is likely to be the most straightforward way of achieving integration, and meet specific requirements. ESBs originated in

[18] Davies, Jeff, Schorow, David, Ray, Samrat and Rieber, David (2008). The Definitive Guide to SOA: Oracle Service Bus, 2nd ed.Springer-Verlag, New York.

response to the need to avoid multiple point-to-point integrations, as the point-to-point approach, whilst clearly useful in specific instances, does not scale well. In the context of the CLIF problem space, then, does an ESB have the right characteristics to merit moving away from a point-to-point approach?

5.1 Characteristics of ESBs

Loose coupling: By abstracting out the ways the systems communicate with each other the coupling between them is loosened. Their location is also transparent, with any one system not needing to know where the other systems are. This allows for client systems in the overall architecture to be replaced or amended without having a major knock-on effect in how they communicate, useful as upgrades take place or individual systems become obsolete.

Mediation: An ESB is designed to serve the client systems it is being used by. Hence, when a client system makes a call to the ESB this may result in a number of actions taking place depending on the services the ESB offers and the content of the message sent by the client. Getting the ESB to do the hard work behind the scenes on behalf of the client allows this to focus on what it is there for.

Service aggregation/orchestration: As part of carrying out actions for the client system an ESB may aggregate and/or orchestrate these to enable them to occur in a particular sequence and according to some conditional logic.

Load balancing/monitoring: An ESB can both monitor and adjust the way messages from client systems are dealt with, and be monitored, to ensure that the ESB is carrying out its role effectively.

For all these characteristics the ESB is adding value to the way systems communicate and work together. The individual client systems could carry out these roles through point-to-point integrations separately. Where each client system is doing the same tasks on multiple occasions, though, there is a lot of sense in getting a common layer to do this work instead.

By contrast, ESBs also have a number of disadvantages:

They work best where there is an associated enterprise model, requiring a full understanding of the overall model for how the systems will work together. Whilst of benefit in itself generating this can be a complex task of its own.

They can result in extra overhead and increased latency caused by messages having to be processed by the extra ESB architectural layer.

In other words, ESBs require effort to implement, and the work required for this plus accounting for the impact of the overall architecture on performance needs to be balanced against the benefits the characteristics bring and enable.

5.2 ESB versus messaging

A key element of an ESB is the messaging, the ability to receive messages, route them, manage them, and carry out the tasks requested by the messages. There are a number of messaging standards available, for example, Java Messaging Service (JMS), and a number of implementations of this standard, for example Apache ActiveMQ, which is used by Fedora. As the introduction of an ESB is largely concerned with improving the communication of messages between systems, it is

beneficial in considering an ESB to assess what the difference is between the messaging implementations on their own and an ESB.

The Apache ActiveMQ website contrasts this messaging implementation with Mule[19], a widely used open source ESB. Mule offers a programming model to support integrations. In focusing on this Mule maintains the ability to make use of separate messaging capability, and hence can be used with ActiveMQ. The model that Mule offers, though, extends what can be achieved simply by messaging, including orchestration and full web services support. It thus puts into practice a number of the ESB characteristics indicated above.

ActiveMQ, in contrast, is purely focused on enabling messaging between systems. Having a clear model for how messaging on its own can support system integration, though, introduces a degree of abstraction that helps to achieve the first of the characteristics listed above, loose coupling to some extent. Hence, whilst an ESB can provide clear advantages and functionality if the effort to implement it can be achieved, there may be potential for using messaging in a more lightweight way.

5.3 ESBs and CLIF

An initial analysis of the potential role of ESBs within the CLIF project appeared to offer considerable advantages. A stated aim of the project is to enable its outputs to be as usable by others as possible. The three systems being investigated by the project are quite different, and developing point-to-point integrations seemed to offer useful if limited scope for others beyond the project. Hence, having an abstract layer through which messages, including content, could be transferred between the systems offers much.

In reality, though, a number of factors have led us away from overt use of an ESB within CLIF. At both partner sites the three systems being investigated represent only a proportion of the system environment across the Universities overall. Hence, instigating an enterprise model to inform the implementation of an ESB was considered unfeasible. Whilst it would be possible to develop an enterprise model on a limited basis, the need to enable the outputs from the project to be used beyond the project affected the capacity to which such a limited model could be used over time. The specific needs of managing integrations between the component systems are also factors in considering the merits of using an ESB.

5.3.1 Fedora – Sakai

CLIF has picked up the original work of the CTREP project to further the development of code that seeks to embed Fedora within Sakai as the default content store. This approach to moving content between the two systems makes extensive use of the Sakai Content Host Handler. There have been instances of where Sakai has been used in tandem with an enterprise ESB, though this has not been widespread. The existence of the CTREP code has made it sensible to pursue this particular point-to-point integration, albeit one that is flexible to other installations of Fedora and Sakai.

5.3.2 Fedora – SharePoint

The integration between Fedora and SharePoint has raised a number of challenges, discussed elsewhere in this document. The different technical origins of the two systems both suggests and

[19] How does ActiveMQ compare to Mule, http://activemq.apache.org/how-does-activemq-compare-to-mule.html

resists the use of an ESB. On the one hand, integrating a Microsoft and Java system using an abstracted layer to carry out the communication seems very sensible: in contrast, finding an implementation of an ESB that can work in this environment, whilst feasible, poses its own challenges. Investigations have highlighted the potential benefit of using messaging between the two systems, though, and for this ActiveMQ is being explored more fully.

5.4 ESBs and the digital content lifecycle

Notwithstanding the specific issues of whether an ESB could, or should, be used for the CLIF project, there remains the broader scope of whether an ESB can assist with the management of the digital content lifecycle across different systems. In this respect there is much that is in favour of using an ESB. Abstracting the communication suggests that whenever content should be moved from one stage/system to another the ESB can mediate this: likewise actions on content as part of its lifecycle could be carried out via the ESB to minimise the impact on the hosting system. An ESB could be considered the common thread that encompasses and supports all stages of the content lifecycle.

To consider the characteristics of an ESB in the context of the digital content lifecycle:

Loose coupling: As content moves between systems it is reasonable to assume that new and/or different systems could be introduced into the system environment being managed. As such, loosely coupling the systems to allow for changes via an ESB would have clear benefits.

Mediation: The degree to which mediation would be of benefit in serving the client systems involved in the digital content lifecycle will depend on the use cases involved and the specific actions required in the lifecycle management. Simple use cases would not necessarily benefit from adding any ESB complexity, whereas complex use cases involving actions to the content may benefit from having a system mediate these. The BPEL processes used in the REMAP project enabled such mediation without being a full ESB implementation.

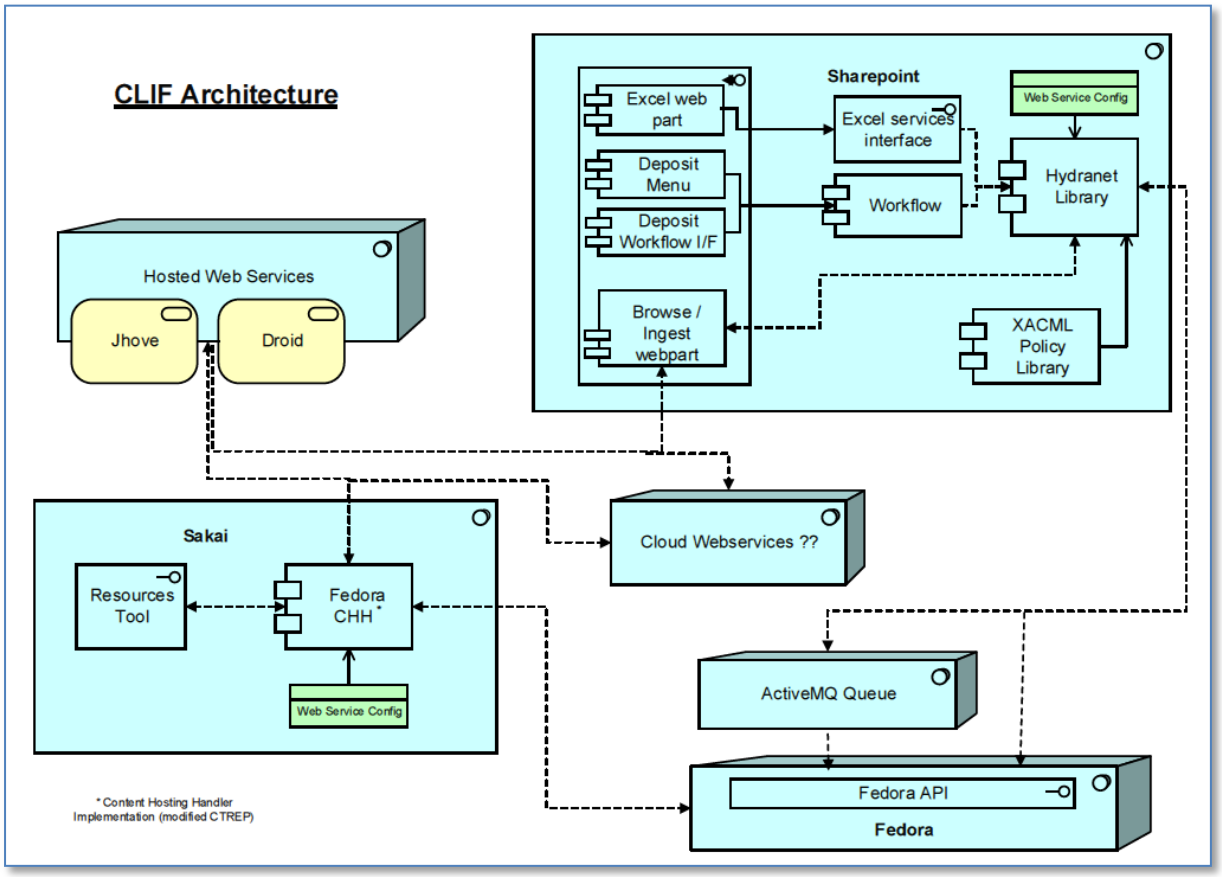
Service aggregation/orchestration: Similarly, BPEL was used to orchestrate a number of services acting upon content being moved from a private to a public repository within REMAP. Where a sequence of actions is required orchestrating of those actions is necessary.

Load balancing/monitoring: The ability of an ESB to monitor processes and be monitored offers a mechanism to manage the digital content lifecycle as an entity rather than as a series of stages.

5.5 Conclusions

Whilst noting the clear potential for using an ESB to support the digital content lifecycle, the practicalities of the systems integration within the system environments of the partner sites have led to the conclusion that implementing an ESB for the purposes of the CLIF project is not realistic. Notwithstanding this there are elements of an ESB that could clearly prove of benefit to the work of the CLIF project, notably messaging. As such, the use of a messaging broker, ActiveMQ, will be investigated with a view to identifying both its benefits and its potential for supporting future integration of the CLIF outputs with a wider ESB environment where this is practically feasible.

6. Technical architecture



The diagram shows the proposed architecture (as at June 2010) for the CLIF Project. Note that both the SharePoint and Sakai systems have the ability to call upon external services such as JHOVE or DROID to provide additional metadata for inclusion in a Fedora object.

7. Acronyms and abbreviations

API	Application Programming Interface
API-A	The Fedora access API
API-M	The Fedora management API
CLIF	Content Lifecycle Integration Framework
CMA	Content Model Architecture (a Fedora term)
CMS	Content Management System
CTREP	Cambridge Tetra Repositories Enhancement Project
DC	Dublin Core – a particular form of metadata
ESB	Enterprise Service Bus
Fedora	Flexible, extensible, digital object repository

	architecture
FOXML	Fedora Object XML – a form of XML used for encoding Fedora repository objects
HEI	Higher Education Institution
HTTP	Hypertext Transfer Protocol
JISC	The Joint Information Systems Committee
JMS	Java Messaging Service
LMS	Learning Management System
METS	Metadata Encoding and Transmission Standard
MIME (-type)	Multipurpose Internet Mail Extension
MOSS	Microsoft Office SharePoint Server
PDF	Portable Document Format
PID	Persistent Identifier
QA	Quality assurance
RELS-EXT	A Fedora object reserved datastream for dealing with external relationships
REST	Representational State Transfer
SOAP	Simple Object Access Protocol – although use of this expansion is now depracated
SWORD	Simple Web-service Offering Repository Deposit
URI	Universal Resource Indicator
URL	Universal Resource Locator
VLE	Virtual Learning Environment
VRE	Virtual Research Environment
WSS	Windows SharePoint Services
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Lanmguage Transformations