Symplectic Repository Tools: deposit lifecycle with interchangeable repositories

Jones, Richard

This paper presents the development of software at Symplectic Ltd to provide a full CRUD (Create, Retrieve, Update, Delete) interface for a number of digital repositories using common, open standards: Symplectic Repository Tools [1]. The primary objective of this work has been to
integrate a research management system (Symplectic Elements [2]) with both DSpace [3] and Fedora [4] (specifically, for the University of Oxford)1, such that the academic's experience of managing their repository content is simple, straightforward and in no way diverting from the overall process of managing their research outputs. The consequences of this include increased uptake of the repository and higher throughput of fulltext content.

Recent work in deposit technology for repositories has tended to centre around the use of a repository as a 'final resting place' for some research item. It has typically used packages of content, roughly analogous to the SIP (Submission Information Package) in OAIS [5], to insert 'finished' works into the archive. An example of this is SWORD [6], which addresses in great detail the deposit mechanism, but is largely reliant on the payload being a single file (for example, a zip), containing all the information that the repository needs in order to create an archival object.

This has a number of negative side effects for the depositor:

- They must definitively make an assertion that some set of content objects represent a 'finished' work. But it may be that a work needs, instead, to be taken in snapshots, over a period of time.
- They must store and manage the content in an organised way before it is inserted into the repository. Since this is a job that repositories are designed for, this duplicates functionality needlessly.

Another problem with the packagebased approach is that describing the content of packages and the formats of any internal metadata files is difficult, and determining whether a given repository is capable of ingesting the package is hard to know in advance. It would be a significant step forward to be able to deposit files individually into the repository's workflow, and to therefore allow the repository to determine when a snapshot of an object is ready to be placed into the archive, while providing stable file storage and management throughout the process. If this is done in concert with a well understood protocol such as AtomPub [7], some of the issues with regard to metadata interoperability can be reduced or even resolved.
The central theme of this paper will be to look at a case study of implementing repository deposit by integrating DSpace and Fedora with Symplectic Elements, where SWORD, and more generally AtomPub (of which SWORD is an extension) are core technologies. The aim was to produce a generalised web service which could enable the integration of these two types of system, which have dramatically different information models, and to support and enhance content acquisition for DSpace and Fedora instances. This included the need to provide a simple, straightforward user interface which would insulate the academics from the underlying complexity.

We will start by looking at the AtomPub protocol and Atom data model and see how they offer a suitable RESTful approach to a full set of CRUD operations for the repository. AtomPub is designed to work using the standard HTTP operations: POST, GET, PUT, and DELETE to achieve this, so we can demonstrate that this is both suitable for use on a webbased repository and also that the standard architecture of both repository systems make it easy to implement such an interface.

We will also briefly discuss the additional reasons that we chose AtomPub over the standard Fedora SOAP and REST APIs, and show that:

- The level of granularity of these APIs is too fine to be used trivially inside a 3rd party
- system, and encourages too strong a binding between them;
- That for Symplectic Elements to be able to integrate with more than one type of repository it
- would be necessary to adopt a common, open standard, to enable portability;
- That the Fedora REST API can be wrapped by an AtomPub API with a reasonable degree of success.

We will also compare the Atom data model to the DSpace and Fedora data models and show that the mapping between them is broadly appropriate. Atom describes its content in terms of feeds and
entries in those feeds, which map neatly onto items and bitstreams in DSpace, and objects and datastreams in Fedora. This allows us to PUT new files (entries) onto an existing item (feed) or allows us to GET feeds which describe the detailed internals of an item.

The combination of these two technologies means that we can Create, Retrieve, Update and Delete items and their files in much the same way as we would with any other web content which works with Atom – for example, blog postings (for which the standard was originally developed).

We will go on to discuss the design and implementation decisions for both the DSpace and Fedora implementations of this standard, looking at what it means to POST a new item into the archive, what the consequences of PUTting a new file onto an existing item are, and how DELETE is handled. It will be of particular interest to compare and contrast the two repository systems and the way that they behave under these conditions, and we will attempt to draw out some lessons that they might learn from each other.

We will look at how the DSpace prepublication workflow, consisting of the workspace and the three stage workflow, should react to changes to the item throughout its lifecycle. This will lead us on to discuss how Fedora, lacking these higherlevel features such as a prescriptive workflow or user
interface, can be compared or interchanged with DSpace under the Repository Tools interface.

Considerable time has been spent analysing repository workflows in general, and an implementation which is capable of taking the following conditions into account has been developed:

- Whether metadata updates should be considered as archivally significant events;
- Whether file deletes constitute a 'rapid takedown' request from the archive, or are required to pass through normal workflow checks;
- Whether academics are required to grant the repository any rights before the workflow can proceed or complete;
- Whether it is ever possible to physically delete content from the repository;
- At what point a new version of an item should be created.

A live demonstration of Symplectic Elements interacting first with DSpace and then with Fedora via Symplectic Repository Tools will be used to walkthrough a deposit and file management scenario which exposes the above issues. We will go through the DSpace demonstration first, and then flip a switch inside Symplectic Elements and run through

exactly the same process using Fedora, demonstrating the interchangeability of the standard and the set of behaviours that it applies to the repository, as follows (using terminology from DSpace):

1. A file is uploaded through the Symplectic Elements interface to the repository; an item is created in the workspace;
2. Rights to disseminate the work are granted to the repository; the item is moved from the workspace to the first stage of the workflow, and the terms and conditions are attached;
3. The item passes the first workflow stage and is moved onto the second;
4. A new file is added to the item; it moved from the second workflow stage back to the first;
5. The item is approved for archiving, and thus completes the workflow;
6. A new file is added to the item, and the original file is deleted; a clone of the item is created in the workflow with the new file – the archive copy will remain unchanged.
7. The new version of the item is approved for archiving, and it is moved from into the archive

– this will create a new version of the item, and the original archive copy will continue to remain unchanged.

We will conclude by claiming that a comprehensive set of web services are required to meaningfully interact with repositories from 3rd party systems, but that the open standards to achieve this already exist to a large extent, and are just lacking in appropriate implementation. We will list some limitations of the AtomPub standard, and how we have felt it necessary to extend this, particularly in the area of adding custom XML to Atom feeds and entries. We will also list some of the extensions and liberties that we had to take with the SWORD standard so that we could break the dependency on the package and move towards individual file deposit and the use of HTTP mimeboundaries. As active participants in the development of the SWORD standard we will ultimately be able to pass
these enhancements on to the rest of the community.

The hope is that this work will show a way towards a broader approach for integrating research management and research information systems with repositories (both open access and closed), such that they can benefit from each other. Further to that, we also expect that this technology could find uses in other 3rd party systems which we have not yet considered, given that we now live and work in an increasingly integrated and distributed information environment.

[1] Symplectic Repository Tools:
http://www.symplectic.co.uk/products/repositorytools.html
[2] Symplectic Elements: http://www.symplectic.co.uk/products/publications.html
[3] DSpace: http://www.dspace.org/
[4] Fedora Commons: http://www.fedoracommons.org/
[5] OAIS: http://public.ccsds.org/publications/archive/650x0b1.pdf
[6] SWORD: http://www.swordapp.org/
[7] AtomPub: http://bitworking.org/projects/atom/rfc5023.html
[8] ORA: http://ora.ouls.ox.ac.uk/