From Dynamic to Static: the challenge of depositing, archiving and publishing constantly changing content from the information environment

Richard Jones, Head of Repository System, Symplectic Ltd
richard@symplectic.co.uk

A repository used for storing and disseminating research is a canonical example of a system which strives to produce stable artefacts which can be reliably referenced, if not actually preserved, over time. This is a difficult task, since the normal state of information is constant flux: being updated, revised, re-written, removed and re-published.

Recent work in deposit technology 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 [1], to insert 'finished' works into the archive. An example of this is SWORD [2], 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 places a burden on the depositor to make an assertion that an item is finished and ready for archiving, and pushes tasks that the repository is traditionally good at (i.e. storing content) out to whatever system the user is creating their work in.

Over the past year, Symplectic Ltd [3] has attempted to break down this reliance on the “package”, and move repository deposit in the direction of not only full CRUD (Create, Retrieve, Update, Delete), but also to give repository workflows the opportunity to define when a work is “finished” (at least, provisionally). This will give the repository the opportunity to do what it does best (i.e. store content), and to allow the administrators – experts in repositories and archiving – to have a hand in determining whether an item is “finished”, relieving these burdens from the depositor and their research process.

At the centre of this challenge is the problem of how to convert dynamic, constantly changing content in the information environment of the academic into static, stable, publishable artefacts in the repository. This is compounded by the need to do it in a simple, straighforward way which insulates the user from the underlying complexity.

The process of producing a research article in a journal has many stages and content elements associated with it. These include, but are not limited to:

- The article itself, pre- or post- print, peer reviewed and not peer reviewed
- Supporting information; documentation produced throughout the research
- Higher quality images than those in the main article
- Source data, collected or produced during the research

Much of this content goes un-seen, and much of it is produced throughout the research and publication process. The objective at Symplectic was to produce an interface to repositories which could accept all of these content types throughout the publication lifecycle.

- At the point the article is submitted to the journal, the pre-print (un reviewed), and of the other supporting documentation, images and data can be placed in the repository.
- After peer review, the un-reviewed version is removed, and the post-reviewed version is added.
- After publication, the pre-print is removed, replaced with the appropriate post-print.
- At any time during the above process supporting documentation, images and data can be added and removed as the researcher sees fit.

The repository should be able to handle this constant change and variety of data without any problems, which is non-trivial using a fire-and-forget style package depositor.

This paper aims to discuss the general problems associated with mapping dynamic content to static content, showing how the repository can play an active role in the process without compromising their archival integrity (and perhaps even enhancing it). In the process of this, we will look at some specifics of repository technology, and how it can be used to solve the problems. In particular we are interested in:

- How can we use standard repository workflows? Are they a help or a hinderance?
- How do repositories manage versioning, and is it sufficient for our needs?
- How do repository data models handle the potential complexity of the bibliographic, and administrative data that goes alongside the content?

Symplectic Ltd has developed an interface based on AtomPub [4] and SWORD for DSpace [5], EPrints [6] and Fedora [7], and have tackled these problems in each of these software environments. There are remarkable similarities between these systems, and subtle but important differences, which make them better or worse at certain aspects of the problem. We will review the workflows, data models and versioning mechanisms for each of these systems in relation to the problem, and find the following worth remarking on:

Workflows

DSpace and EPrints have formally constructed workflow processes, while Fedora has none. EPrints has a single space (called a "buffer") in which items being validated for archive sit; this makes it more difficult for EPrints to explicitly make the administrator aware of workflow-affecting changes in the item. DSpace, meanwhile, has a multi-stage workflow, and this makes it easier, although by no-means ideal, for the system to alert the administrator when an item they are validating has been changed from outside. Fedora has no such concepts, although it is common for workflow state to be stored inside Fedora objects, giving it significant scope for managing incoming dynamic content, but with an associated amount of effort required on the part of the owner.

Versioning

Fedora has file versioning built-in, but no formal support for higher level "item" versioning; this makes file alterations straightforward, but larger structural changes to the item need to be custom managed. EPrints has item level versioning built in to its user interface, but only supports a single version chain; since it is possible in certain environments to have version chains which bifurcate and merge, this can be a hinderance as much as it is a help. DSpace has no versioning support at any level, although it is possible to "fake" versioning via metadata annotation; this has the advantage of being extremely flexible, but the disadvantage of being completely unrecognised by the application itself.

Data Models

There is broad agreement across the repository platforms as to the appropriate data model. Certainly in terms of structural model, there is barely any difference, although only DSpace recognises multiple ownership for its content elements. Key differences emerge through metadata handling: DSpace has a strictly flat native metadata mechanism, EPrints supports hierarchical metadata, and Fedora is highly agnostic to the process, preferring just to store metadata in files.

By examining these features of the repository, we will show how it is possible to implement a good archiving solution for a changing set of content, and will suggest improvements to all the repository architectures, emphasising the following features:

- That versioning should be on all scales: both files and structurally
- That versioning should allow multiple inheritance and multiple children
- That data models should support hierarchical metadata
- That repositories should include better tools for managing and manipulating complex objects

There is a fair amount of work to do before this technology is broadly available, but we will demonstrate its use in Symplectic Elements [8], a Research Management System, which allows academics to manage their repository content alongside their publication lists, professional web pages, and academic CVs. It is heavily repository agnostic, with implementations for DSpace, EPrints, Fedora and IntraLibrary (currently under development) [9], and will also shortly be extended to DigiTool, a commercial repository from Ex Libris [10]. We will show an idealised walkthrough demonstrating all the aspects of the system: integration with workflow, CRUD, versioning and archival integrity.

We will conclude by claiming that this technology would be of benefit to both researchers and repository managers, by offering to bring them closer together, without either side having to become familiar with eachother's technology environments. The advantage to the researcher is never having to interface directly with a repository, while acquiring “deposit” as a by-product of simple file management. The advantage to the repository manager is an increase of throughput to the archive, and intervention earlier in the publishing lifecycle.

[1] OAIS: http://public.ccsds.org/publications/archive/650x0b1.pdf
[2] SWORD: http://www.swordapp.org/
[3] Symplectic Ltd: http://www.symplectic.co.uk/
[4] AtomPub: http://bitworking.org/projects/atom/rfc5023.html
[5] DSpace: http://www.dspace.org/
[6] EPrints: http://www.eprints.org/
[7] Fedora Commons: http://www.fedora-commons.org/
[8] Symplectic Elements: http://www.symplectic.co.uk/products/publications.html
[9] Intrallect IntraLibrary: http://www.intrallect.com/index.php/intrallect/products
[10] Ex Libris DigiTool: http://www.exlibrisgroup.com/category/DigiToolOverview