Mounting Books Project
Presentation Proposal for Open Repositories 2009

Northwestern University Library
Claire Stewart, Head, Digital Collections
Steve DiDomenico, Digital Repository Developer
2/20/2009

Summary

The Northwestern University Library undertook a software development project to create an automated workflow to enable files from its Kirtas book scanner to be both linked to the OPAC with a page viewer application, and ingested into its Fedora repository as archivally sustainable and reusable digital objects. The web-based Book Workflow Interface (BWI) software utilizes jBPM for management and web services for key creation components. It also features an AJAX interface to support drag-and-drop creation and editing of METS-based book structures. The BWI system ingests locally scanned texts as well as texts digitized by external partners or vendors.

This project addressed the need for a Fedora-based book viewing tool that can be used by other research libraries developing digital repositories based on a Fedora systems architecture. The book view interface includes full-text search and view, search-within-a-book, book structure browse, page turning, and zooming interface components. The workflow system can be expanded over time to support new functions in the book publishing process, and can be redeployed in support of digitization processes for other types of media.

Shifting from a simple book reformatting operation to a dynamic program that makes any multipage text object fully accessible online, this system dramatically improves Northwestern's ability to share its unique library and archival collections. The project was fully supported by the Andrew W. Mellon Foundation and the Book Workflow Interface and public book viewing software will both be released as open source in spring 2009.

Book Workflow Interface (BWI)

The Book Workflow Interface (BWI) is a web-based software program that allows operators to manage their jobs from scanning all the way through ingestion and public display of content. The software uses jBPM (Java Business Process Management) for tracking jobs, EXT-JS for structure-building and object-matching components, and web services to interact with other systems.

When an operator logs into the interface, she is presented with a list of jobs--each job represents a book or volume that is currently in the process of being scanned and ingested. She also has the option of starting a new job.

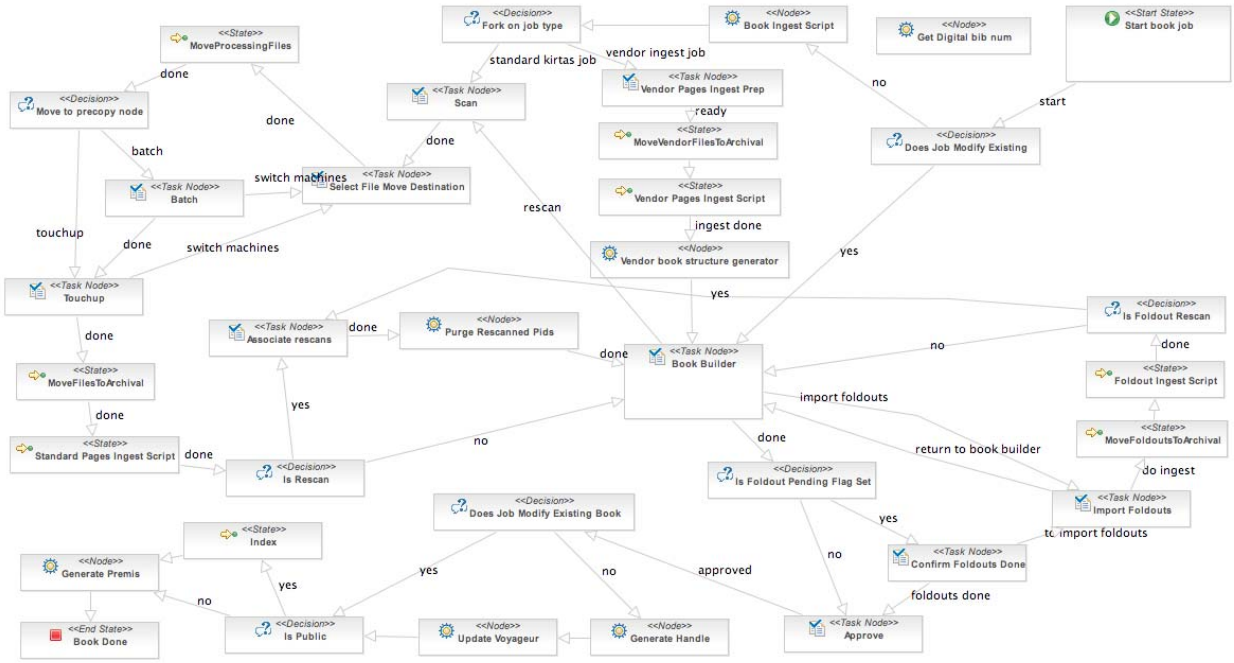
A number of different things happen during each step of the workflow (this is a simplified version):

1. The operator enters either a bib number for the book, or the book's barcode.
2. A Yaz Proxy query is made to the Library catalog to capture DC, MODS, and MarcXML. This metadata is eventually saved in the Fedora book object.
3. The operator selects an option to continue with the job. A preliminary Fedora book object is created. The workflow takes them through scanning and image correction steps.
4. When the images are ready, the workflow automatically copies the files to an archival server. It then triggers an ingest process, where jobs are queued. Image conversion takes place, FOXML is generated for the page objects, and the page objects are ingested into Fedora.
5. The operator is presented with the Book-Building software (see below), which allows them to create book structure and tag pages.
6. The operator can then preview the ingested book, and then approve it.
7. A handle is created for the book; the Book Workflow Interface triggers this handle creation through a web service.
8. The handle URL is saved into an 856 field in the Library catalog via a web service.
9. The search index is updated; the search index uses Solr and also updates through a web service.

There are also some optional steps that can occur in the workflow:

1. If a book has images that need to be rescanned, the operator will be presented with a screen that allows them to associate the new images with the old ones. The Book Workflow Interface then copies the new image's datastreams over to the old ones. Then the new page objects are deleted. This ensures that the old Fedora pids will always remain the same, and any possible future annotations of the object will remain intact.
2. The operator has the option of adding foldouts (images that are captured separately from the Kirtas book scanning processes).
3. At any point the operator can suspend a job if they see a problem. Jobs with these issues won't continue moving along the workflow until the problem has been corrected.
4. Images that were scanned outside of the Kirtas bookscan process can be imported.

The BWI has been built to be very modular; components it interacts with can be changed out if necessary since it uses web services to interact with them. And since the files that are saved from Kirtas are standard JPEG images, a different book scanning system could be substituted with minimal effort. As the workflow changes over time and new steps are added, such article-level markup, the workflow can be easily modified to include these new nodes.



Mounting Books jBPM workflow diagram

Queue server

One of the services that the BWI interacts with is a queue server. The Queue server is notified of jobs that need to be processed, and queues them up to avoid overloading the server with too many intensive processes. For the Mounting Books project, each set of images is converted to JPEG2000, has MIX metadata generated with JHOVE, EXIF data is generated on the original camera images, and three OCR outputs are created (image plus text PDF, plain text, and XML). The Queue server also triggers FOXML generation and ingestion into Fedora.

The BWI queries the Queue server to identify the status of processing. When the processing is complete and the data has been copied to the appropriate final locations, the Queue server reports they are done, and the BWI then knows to continue the workflow.

The Queue server has other opportunities to be used with other projects. Image projects and also audio conversion projects may be able to be to take advantage of this; virtually anything that is being processed in a queue could use this system.

Book-Building Software

As part of the workflow, operators have the opportunity to create structure for their books. The operators can create chapters and other subdivisions; these are represented as folders that other folders or pages can be drag-and-dropped into. Each structure (such as a chapter) can be given a number, and a label as well. These structural elements are then formulated into a JSON query, which gets processed and saved as a METS file in Fedora.

The Book-Building software uses technologies such as EXT-JS, which is a robust JavaScript/AJAX interface library. It also uses JSON for processing data, and utilizes an Aware JPEG2000 server to allow the operators to quickly view their images.

Other features of the Book-Building component include automatic page numbering; operators can select a range of pages and have them automatically numbered descending or ascending. Or, optionally operators can supply their own page numbers. Also, operators can tag pages within a book (i.e., illustration or figure); these tags could be useful in the future for scholarly work or collection maintenance.

Book and Page Object Models

In order to accomplish data ingestion and display, a book object model had to be created for Fedora. The book object model consists of Dublin Core, MODS, MarcXML, METS (for the book structure), and RELS-EXT. The RELS-EXT contains relationship data for the book's pages; the book object is acting as a collection object.

The page object model includes Dublin Core, MODS, RELS-EXT (which contains relationship data for its parent, the book), OCR PDF, OCR plain text, and OCR XML. It also includes the archival versions of the image, EXIF camera data for the original image, a JPEG2000 version, MIX for all variations of the image, and SVG (which can be using for cropping).

The page object model also includes Fedora disseminators, which allow for a consistent interface to image cropping and resizing. This can allow us to change the underlying JPEG2000 server software later on, while keeping the same disseminators and parameters on previous projects.

Public Display Interface

Users from all over the world will be able to access books scanned by the Library. The web-based interface is designed to be easy to use and provides fast, readable access to page images. The pages are delivered using a concoction of AJAX (using the Ext-JS library), JSON, and Fedora disseminators.

Features of the public interface include:

- Full-text search (based on Solr indexing of OCR'ed text)
- Search-within-a-book
- Book structure browse - The book structure comes from the METS markup supplied by the book building software
- Page turning interface - Users can view book pages by a long list of thumbnails, or see them one at a time
- Raw text view - Users can view the OCR'ed raw text instead of the page image.
- Zooming interface - Users can zoom in on images either using the AJAX interface, or using a flash viewer. Currently, both of these use the Aware JPEG2000 image server to supply resized JPEG images on the fly.
- Integration with the Library catalog - The Library's catalog record links to the public display interface, and vice-versa

Future directions

Once the Northwestern Books public site is launched, attention will turn to other extensions

of repository-based services. A member of the Committee for Institutional Cooperation (CIC), Northwestern expects to eventually be able to integrate locally scanned books with books from CIC libraries digitized by Google and stored in the HathiTrust. Along with other types of objects stored in the Northwestern repository, authenticated users must be able to create personal collections and deeply annotate digital objects. Further "uptagging" of raw OCR'ed text to support article-level and other sub-page indexing, part of speech, TEI and other tagging, and text analysis will be explored. Finally, the workflow support system will be extended to support other library-based content conversion and ingestion. In the future, the jBPM system may be extended to support community collections creation and deposit.