A Workflow for Document Level Interoperability

David Bainbridge
University of Waikato
Hillcrest Road
Hamilton, NZ
davidb@cs.waikato.ac.nz

Sally Jo Cunninghom
University of Waikato
Hillcrest Road
Hamilton, NZ
sallyjo@cs.waikato.ac.nz

Abstract *This article describes a software environment called the Exchange Center that helps digital librarians manage the workflow of sourcing documents and metadata from various repositories. The software is built on Greenstone but does not require its use as the final digital library server. After describing the software architecture we provide two scenarios of its use: a private library of recipes, which ultimately involves collaboration with other cooks; and a digital library that aggregates the collections of various host institutions that use different repository software.*

Keywords Digital Library Interoperability, Software Architecture, Workflow

1 Introduction

The term Digital Library (DL) can mean different things to different people. Even within the profession there are a variety of different definitions [15]. This has lead to a sizable—some might say dizzying—array of software solutions for libraries (archives, and others) to choose from, as a gamut of comparison articles attest to, e.g., [7, 6, 3]. Recurring in these articles are the themes of importing, exporting, and interoperability; yet, surprisingly these are areas that are (on the whole) under-represented in the software solutions available.

Building on our experiences with Greenstone [14], this article describes a software environment that we call the Exchange Center, designed to help digital librarians manage the workflow of documents and metadata in forming various types of digital repositories. Tasks include sourcing from external digital libraries, adding local content where necessary, and manipulating the resulting collection into formats that are suitable for importing into other digital library repositories. Such work involves interoperability at a practical level, reflecting the real need of practicing digital librarians to span different standards and also work within different digital library software applications.

This paper develops the work previously reported as an extended abstract [1], and is structured as follows. First we provide an overview of pertinent features of the existing Greenstone [15] infrastructure that supports the Exchange Center. Then we describe the graphical interoperability environment. Key features are the import and export stages, which we describe in detail from the user's point of view. Several protocols are supported for importing from external repositories: HTTP, OAI-PMH [9], Z39.50 [10], and SRU [12]. Export options include DSpace [11], and the Fedora [8] variant of METS [4]; XSL Transforms [13] are used wherever possible to aid extensibility. The paper concludes by describing two scenarios that illustrate how the Exchange Center can be put to use, along with a discussion of issues that arise.

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

2 Software infrastructure

Greenstone is intended to be a flexible software architecture for digital library construction and delivery. Pertinent to the present project is its importing component. This can stand alone—it does not have to result in a Greenstone collection—and includes a system of plugins for processing documents and metadata that is both extensive and extensible. There are plugins for literally dozens of widely-used formats, some of which we draw upon in the scenarios below. The import process combines documents and metadata into the METS format (Greenstone profile).[1] Collections can be exported in other formats, for example DSpace (ready for the DSpace batch import program) [17]. The software includes an OAI server and a Z39.50 server, which are alternative ways of getting collection information out of Greenstone, as well as Greenstone's native Reader's Interface. See *www.greenstone.org* for further details of this software.

On top of this infrastructure is the Greenstone Librarian Interface (GLI) [16]. This is an interactive application that provides a graphical environment for digital librarians and others to gather, enrich, design, and create collections. Written in Java, in its standard deployment it performs its work using the local file network: this is where it finds the documents and where it builds the collection. GLI was originally designed

[1] Historically, because Greenstone predates METS, a canonical XML file format known as the Greenstone Archive format is used by default, but METS can be specified instead using a switch on the import program.

specifically to create Greenstone collections from existing documents and metadata. However, this paper focuses on how we have adapted it to be used to assemble information for other digital library systems too. Since it was designed with flexibility in mind, it is not difficult to re-target it to fulfill this new role of a document and metadata exchange center.

Figure 1 shows the re-purposed graphical environment, which we call the Exchange Center. Along the top are *Download*, *Gather* and *Enrich* tabs. *Gather* refers to the act of gathering together documents and metadata into a digital library collection. It involves dragging individual files, folders, or entire file hierarchies from a file browser and dropping them into the nascent collection. *Download* invokes utilities that bring in documents or metadata over various standard protocols; this is a precursor to the gathering stage for collections in which the user must reach out to other digital libraries—or to the web at large—to access their content. The downloaded files are placed in a cache where they can be examined and dragged into the collection. Finally, *Enrich* is used to add metadata manually to documents in the collection. If there are existing metadata files, they can be dragged into the collection just as documents are. However, users often need to add metadata to documents interactively. GLI is agnostic towards the actual metadata set (or sets) deployed, and facilities exist to define new ones if desired. Exporting the collection in different forms is achieved through an *Export* item on the *File* menu.

The full Greenstone Librarian Interface, in addition to these three tabs, contains two others that take the user through the design and building process for a Greenstone collection. These are not used in the Exchange Center, and consequently are suppressed.

3 Importing from external repositories

This section details how we augmented the existing GLI capabilities to reach out to external digital libraries and repositories. The implementation of this was split into two stages: core infrastructure and graphical environment. The first stage addresses the actual remote information gathering process, while the second stage looks at adding this functionality to the user interface in an extensible way. We start with the latter to convey the end result of how things pan out from a user's point of view, before detailing how the infrastructure supports this.

3.1 Graphical environment

Figure 1 shows the download panel in use. At the upper left is the operation area. This presents a list of protocols that can be used: currently Web, OAI, Z39.50, and SRU. In this case OAI is selected. At the upper right is the configuration area, in which various settings relevant to the selected protocol are available. Here the user has entered the name of an OAI server, *http://memory.loc.gov/cgi-bin/oai2_0*, decided to download a maximum of 200 metadata records from the set *cwp* (Civil War Photographs), and pressed the *Download* button. In the lower portion of the panel is the download area. Here a progress bar has appeared (the lower one), providing feedback about how the download operation is proceeding. The upper progress bar shows the result of importing metadata over the SRU protocol, which was invoked earlier and is now complete.

The **Configuration area** contains four download methods for users to select and a set of configuration options that change depending on which method is selected.

For downloading over the web we use a standard web-mirroring program, which is controlled through a form-based panel. There are two arguments, *url* and *depth*. The first must be specified; it gives the web location from which pages will be downloaded. The second is optional, and specifies how many levels of information are to be obtained. For example, if *url* points to a particular home page and *depth* is set to 0, the system will only download that page. However, if *depth* is set to 1, then the system will download the home page as well as all the pages within that site that the home page links to.

We have already examined the OAI mode, which is selected in Figure 1. As well as the (compulsory) *url* argument and the (optional) *set* and *max_records* arguments, which have already been seen in use, there is a further argument, *get_doc*. This determines whether the user wishes to download the source documents along with the OAI records. Although OAI-PMH is a metadata-only standard, many users follow the informal convention that the *Identifier* field, if present, gives the URL of the document to which that metadata refers. From the point of view of the destination digital library, a useful source for full-text indexing. Setting the *get_doc* flag causes them to be downloaded also and bound to the metadata.

Settings for the Z39.50 mode are slightly more complex. There are four compulsory arguments and one optional one. The *host* and *port* arguments are used to specify the location of a Z39.50 server. The server might contain more than one database, and the *database* argument specifies which one to access. The *find* argument allows users to input the keywords they wish to search for and also supports advanced query syntax. Finally, users can enable the optional argument *max_records* to specify the maximum number of records to download. SRU follows a similar route. It has exactly the same options, with the server interchange realized through the same underlying open-source utility, YAZ (see *www.indexdata.dk* for more information).

The **Operation area** lies directly beneath the Configuration area in Figure 1 and contains three buttons. *Download* starts the actual download operation according to the setup in the configuration area, and adds a

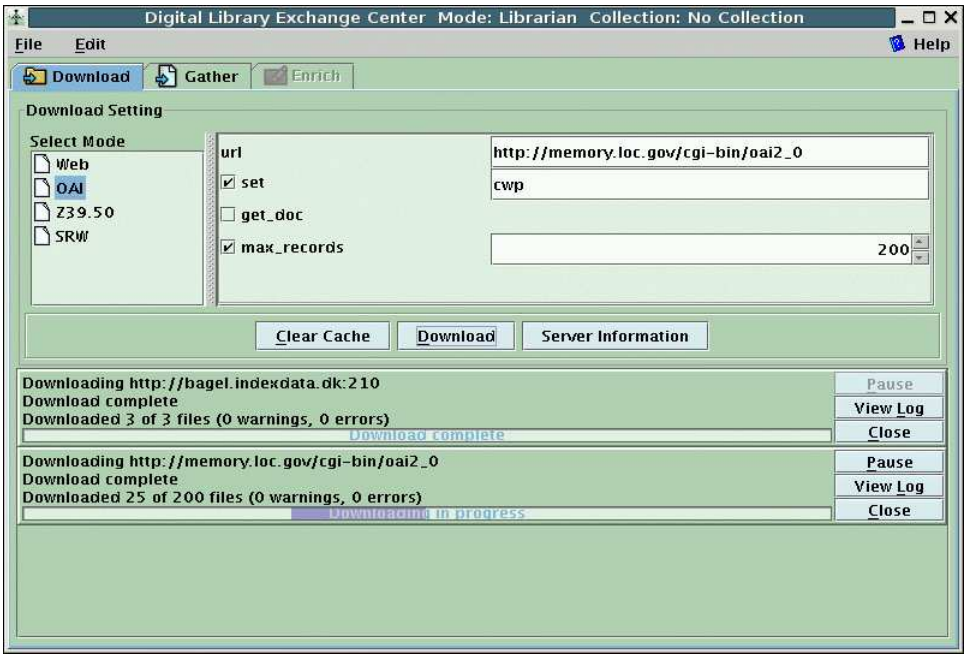

Figure 1: Exchange Center interface – Download Panel.

new job to the progress area. The files are placed in a folder called *Downloaded Files* which is present in the Gather panel, allowing users access to them. This panel is shown in Figure 2. As mentioned previously, the user drags files into the collection. The file system is presented on the left-hand side. At the top level is the user's local file space and home folder (second and third entries), documents in existing Greenstone collections (first entry), and downloaded files (fourth entry), which is shown expanded in Figure 2.

Returning to Figure 1, the *Clear Cache* button deletes all the information in the *Downloaded Files* folder. Finally, the *Server Information* button provides some basic information about the specified information source, which is helpful in targeting a particular set on an OAI server, for example, or determining the databases that exist on a Z39.50 server.

The **Download progress area** shows all the download jobs that have been issued so far. They are stored in a process queue, and each one is processed in turn. For each job there is an associated information bar, beside which are three buttons. The information bar displays the status and progress of the current download job.

Downloading proceeds asynchronously with the rest of the interface. While the application is busy working through the download queue importing data files, the user is free to interact with the other tabs in the interface. For example, he or she can begin to gather files into the collection for processing by the Exchange Center, or enrich the metadata of files that have already been included.

Three buttons are visible in Figure 1: *Pause*, *View Log* and *Close*. The first temporarily stops the download process, and yields priority for downloading to the next download job. *View Log* displays the log information for the current download operation, which gives a detailed account of progress. *Close* deletes the current download job and removes it permanently from the panel. Any files that have already been downloaded will remain in the *Downloaded Files* folder.

3.2 System Description

This interface relies on Greenstone's core infrastructure to perform the actual download operations. Figure 3 shows a flow diagram for the *Download* panel along with the corresponding core infrastructure scripts for supporting the download related processes. Once the Download panel is initiated by users, the system defaults to Web download mode. The arguments are obtained by using the command-line script *downloadinfo.pl* described below, and the system will use the obtained information to generate the GUI components in the configuration options area of the Download panel. This technique is used to aid extensibility.

Once a mode is selected, users need to fill out the compulsory options in order to retrieve the server information or download the actual documents. By pressing the "Server Information" button, the system retrieves some basic information from the information source. This process is supported by the core infrastructure *downloadfrom.pl* script with the "-info" flag. By pressing the "Download" button, the system starts the download proper (by running *downloadfrom.pl* without the "-info" option).

3.3 Core infrastructure

The Greenstone building infrastructure consists of a set of utilities, written in Perl, for building, unbuilding,

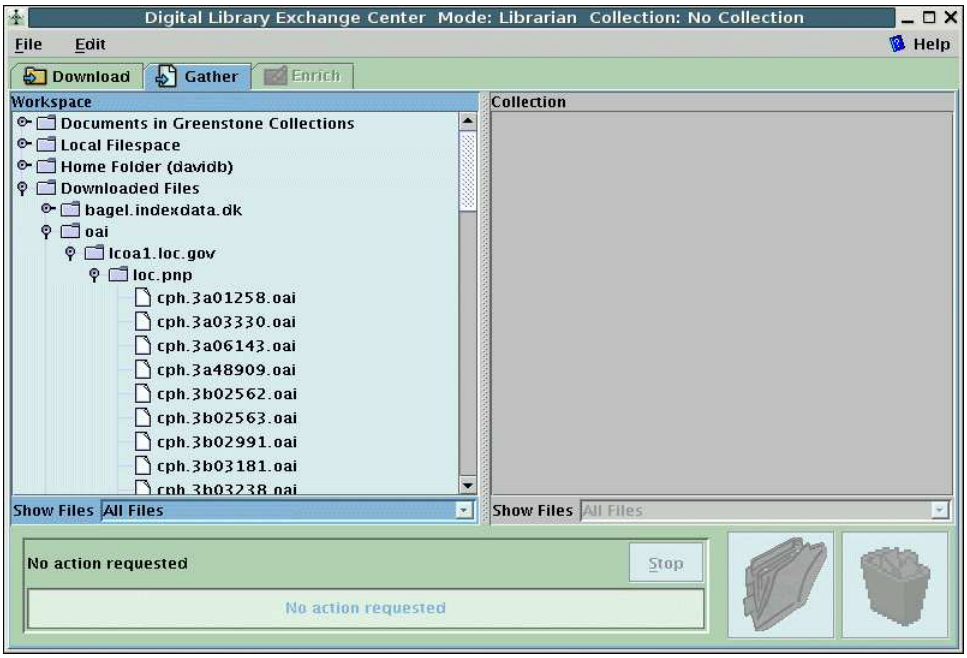

Figure 2: Gather Panel – downloaded files.

creating CD-ROM images, and the like. An existing command for downloading XML records from an OAI server already existed and this was generalized to become *downloadfrom.pl*.

Much of the work for *downloadfrom.pl* resides in a set of Perl modules. These modules use an inheritance hierarchy to abstract to the core what it means to contact and download documents and metadata from other digital libraries, thereby paving the way for further classes of protocol, such as that used by Fedora to be added. We return to this point in the conclusion.

4 Exporting for external digital libraries

For the export ability of the Exchange Center development, modest upgrading of the existing software was undertaken. Figure 4 shows the process of exporting a collection, through the graphical environment, to the Greenstone profile of the METS. The dialog is initiated by the *Export* option of the *File* menu. In the resulting popup window, users can select the format—currently a choice between DSpace and METS—and the collections they wish to export. Optionally, the user can stipulate an XSLT file that is applied to the generated file. This opens up the possibility of generating file formats other than the two explicitly mentioned. For instance, an XSLT of modest complexity can transform the Greenstone METS format into Fedora METS.

Compared with the plugin architecture that is the powerhouse of the importing process, the exporting infrastructure is less developed. Figure 5 shows the process in schematic form. At its heart, the internal document model—generated as a consequence of importing—is traversed by a "back-end" that generates the necessary files. The *docsave.pm* module is the initiation point for this. There are different back-ends for the different formats supported. This means in principle a new traversal routine must be added for each new export format supported. The ability to apply an XSLT softens this requirement slightly, however, and provides some degree of extensibility. While a back-end for DSpace had already been written, as a proof of concept an XSLT was written to demonstrate how it could be done without modifying Greenstone's core Perl infrastructure, mapping the Greenstone METS profile into that used by DSpace. These back-end modules are implemented as a system of "plugouts"—a complementary process to plugins, that provides a rich and versatile way of exporting collection content.

5 Scenarios

To illustrate different uses of the Exchange Center we describe two scenarios.

5.1 Web to Fedora

George is a enrolled in a Masters of Library Information Science programme, currently taking a course in information management. For one assignment he must organize a personal collection using one of the software digital library packages they have studied in the course. Since George loves to cook, he chooses this pastime as the basis for his assignment.

When George cooks, he uses a variety of resources: cookbooks, recipes he has jotted down on scraps of paper and compiled into a scrapbook, and recipes that he finds and normally bookmarks from the web. For this assignment he decides to organize all this content using Fedora as the document repository.

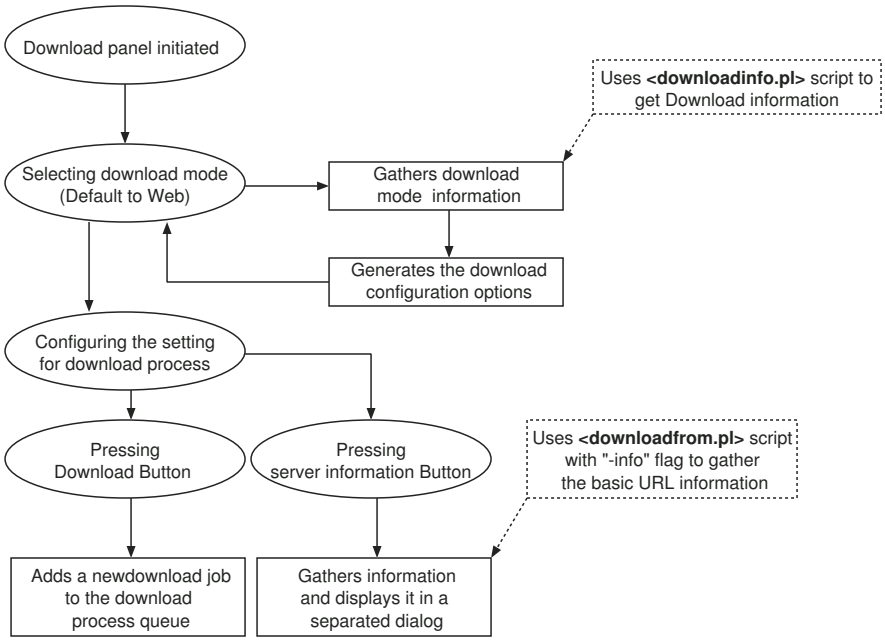

Figure 3: System flow diagram for Download Panel.

He begins by digitizing the pages of cookbooks that contain recipes he has used at least once. Sometimes he expands his policy and digitizes a popular section entirely. He types in the recipes from his scrapbook as plain text files, performing a modest amount of formatting with tabs and line breaks. For example, the first line of the file is the recipe's title. He uses the Exchange Center's *Gather* panel to get these page images and files into the collection, and then the *Enrich* panel to add a small amount of metadata. He does not need to enter title information for the text files, because these are automatically determined from the first line of text.

For the web recipes he retrieves the items using his bookmarks and, where needed, Internet searching. Using the *Download* panel he selectively downloads individual recipes, recipe index pages plus the recipes they point to, and, when the case merits it, entire websites. All these he drags from the *Download Folder* into the collection, except that he takes the opportunity to tidy up as he goes, deleting chaff such as preface material and other non-recipe pages. Again he assigns metadata manually using the *Enrich* panel, providing a unifying structure for all the digital objects in his collection. He exports the collection to Fedora and starts to use it.

Unbeknownst to George, his friend Mildred who is also taking the course, has begun a similar project with her recipes; however, she has chosen to use DSpace. When they realized what each other had done (how they laughed!) they swapped website addresses of their repositories with a view to incorporating a selection of tasty morsels from each other's collection.

Mildred searches and browses George's collection and contacts him with a list of recipes she would like, identified by their unique IDs. George uses this information to start a new Exchange Center collection into which he drags and drops the recipe files from his main collection. When copying a document from an existing collection, the Exchange Center automatically transfers the metadata too. George then chooses the *Export* option from the *File* menu, selecting DSpace as the export format. If there was a mismatch between the metadata fields he had used and those Mildred had chosen, he could have reconciled the differences through an XSLT. The exported DSpace files are then zipped up and e-mailed to Mildred.

To transmit some of Mildred's records to George, they use OAI-PMH to facilitate the transfer. This involves a similar workflow to the next scenario, where the Exchange Center is used to import from a federation of OAI repositories, consequently we defer describing this particular transfer of information until after the second scenario.

Discussion

George and Mildred achieved their goal with a small amount of effort, despite the fact that they had not come up with a joint plan in advance. The procedure was a little less than elegant in that when Mildred chose some of George's recipes, she found them using Fedora's Reader's interface and then George had to copy them from the collection using the Exchange Center interface. This would have been easier if the environment had included a "berry basket" facility (akin to the popular eCommerce metaphor of placing items in the shopping cart) that allowed Mildred to select juicy documents and export them, for example by email. But nevertheless the procedure they devised was practical, and worked reasonably well.

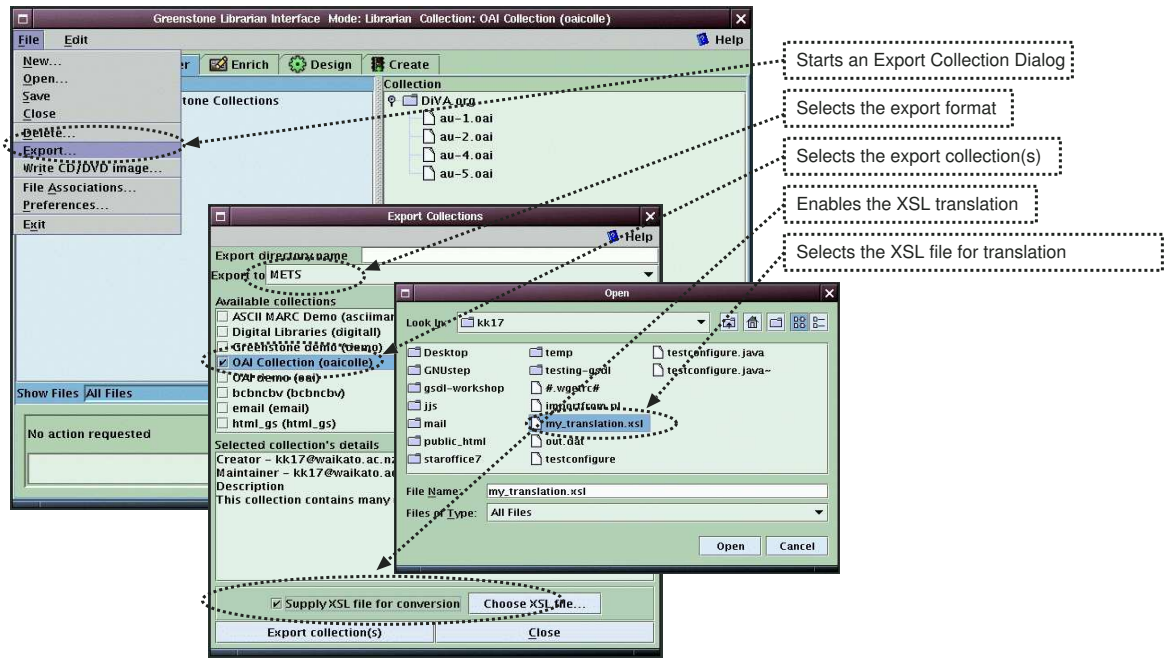

Figure 4: Graphical exporting process.

5.2 An OAI-based institutional repository network

A national consortium of university libraries has decided to establish a network of institutional repositories. Each university sets up and manages its own repository. It is free to choose its own software solution, with the proviso that it must support OAI-PMH, with MODS metadata [5], and a persistent URL to the source document in the *Identifier* field. DSpace and Greenstone are popular choices for the individual libraries. DSpace has a web interface through which people can enter new documents; Greenstone supplies similar functionality using a submission system called the Depositor [2] in combination with an applet version of GLI for building and maintaining collections on a remote server. The plan is to create a single website for aggregated searching and browsing once the individual institutions' sites are up and running.

The Exchange Center described in this paper is chosen as the interoperability environment in which to develop the aggregated site. The *Download* panel is used to import OAI records from the various university servers to a central location on the file system (the download folder). These are then copied across verbatim into the collection. One of the Exchange Center's many plugins is an OAI plugin which reads OAI records and associated documents, if any, and it is used to form the collection. However, the OAI plugin supports Dublin Core metadata, not MODS. Fortunately all XML-enabled plugins have a switch that allows the user to supply an XSLT transform, and this is used to provide a crosswalk from MODS to Dublin Core. Guidelines from the Library of Congress are used in establishing the mapping. Some information is lost, but that is not important in this context because the individual institutions are responsible for archival copies of their documents and metadata; the aggregated site is just to provide unified access to all the information.

Irrespective of which university it originated from, the Dublin Core version of the metadata is stored locally at the central site. This makes it easy to manipulate and display. However, the source documents themselves reside on the relevant host institution's server. It is those servers, not the central one, that will serve the actual documents from the aggregated collection to users. Some form of bridging link is required. Using standard features of the Greenstone system, an *Source_URN* metadata field can be established that contains each document's location. It is easy to incorporate this into the web pages served up by the system—for example, by hyperlinking the title—in such a way that the document is downloaded from the relevant host institution should the user click on the link.

Discussion

If full-text indexing search were desired for the aggregated document collection, the *get_doc* feature would need to be activated. Recall that this makes the download code check each record's *Identifier* field and, if it is a valid URL, download the file and tie it to the record as its source document. This is all that is needed for the collection that is built to provide full-text searching. Furthermore, if desired a cached version of the document could be offered to readers as an alternative to the host institution's copy in case its server went down.

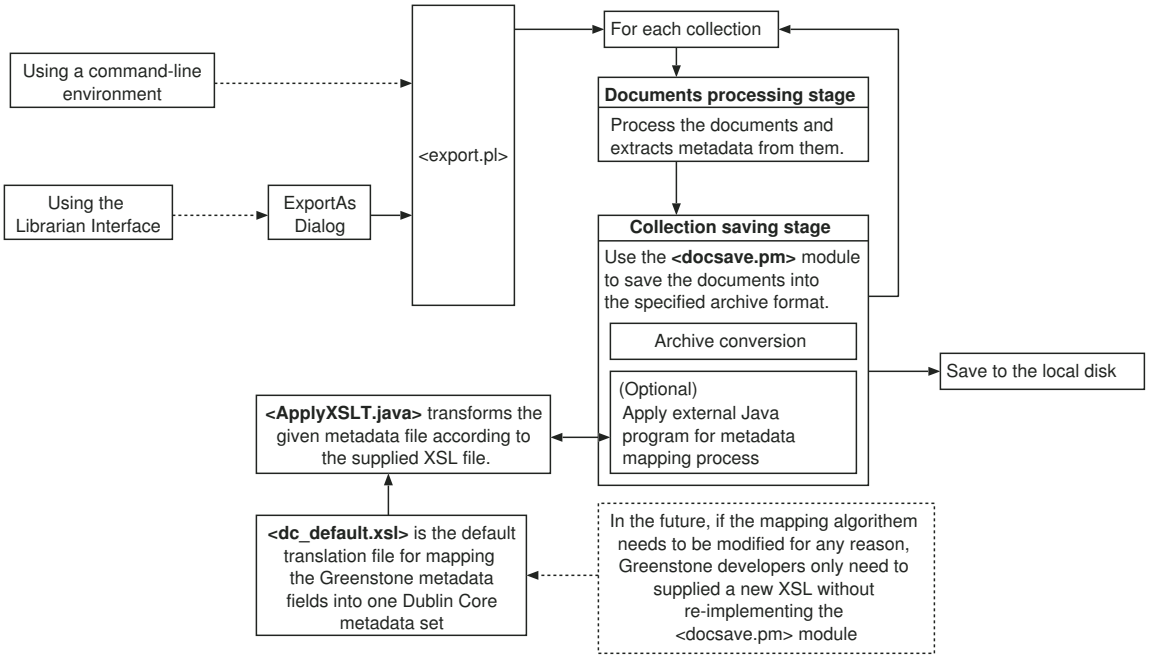

Figure 5: The exporting process.

In moving to the production version of the aggregated site, manual interaction using the *Download* panel would be discontinued in favor of setting up a script that runs the relevant Perl script (*downloadfrom.pl*) automatically every night. A finishing touch for the project is to brand the various host sites, and the central server, with the appropriate institutional logos, backgrounds, and color schemes. This is straightforward to accomplish in both DSpace and Greenstone.

As an alternative, it might be desired to present the aggregated collection using DSpace rather than Greenstone. This would involve using the Exchange Center to accumulate the documents and translate the metadata into Dublin Core. But instead of building the collection with Greenstone, the *Export* option would be used to export it in a form that is suitable for DSpace's batch import program. The above remarks about downloading the full text for indexing apply equally in this scenario, and of course the production version of the system would use automated scripts to perform the exporting and importing into DSpace.

Now we have described the basic approach the Exchange Center uses for OAI repositories, we can now return to the role of this protocol in the recipe scenario. In many ways the workflow for importing recipes from Mildred's collection into George's one is simpler: there is only one site to connect to and one download session is enough to transfer the metadata and source documents (using the *get_doc* option). But here's the rub. To achieve the selective inclusion of recipes accompanied with the assigned metadata, the OAI download step needs to download *all* the metadata records and *all* the source documents, even though only a small portion will ultimately be selected. It would be better if only the metadata records were requested (i.e., no *get_doc*). This metadata can then be used to choose what goes into George's collection and it is only when the collection is built that the source documents are retrieved. This would necessitate that the OAI plugin supports the *get_doc* option as well, and while not difficult to arrange it has not been done to date.

6 Summary

In summary, we have described an interactive system, the Exchange Center, that is intended to help practicing digital librarians solve practical interoperability problems. It is based on the Greenstone infrastructure, but is by no means restricted to projects that build Greenstone collections. The software already has an extensive and mature infrastructure for information gathering (including from external sites), enriching downloaded documents with manually-entered metadata, and processing documents in a wide variety of formats, including image, audio, and video formats, into a standard form. This project makes this infrastructure available to users of other digital library systems by permitting the collections to be exported in various forms.

Information can be downloaded over HTTP using standard web mirroring facilities, from OAI metadata repositories (including the documents themselves where applicable), and from Z39.50/SRU library servers. Of course, documents and metadata in local files can also be included in collections. While Greenstone's common internal format predates the design of METS, an option allows users to specify that METS should be used for storing the collection instead. This facilitates greater use of existing

standards. Collections can be exported using an OAI server, a Z39.50 server, in various METS profiles (using XSLT translation modules), and in a form suitable for DSpace's batch input process.

The system is by no means restricted to the current set of input and output protocols and formats. Extensibility is absolutely fundamental to Greenstone's design, and the Exchange Center inherits this philosophy. By implementing and demonstrating four very different protocols, HTTP, OAI-PMH, Z39.50, and SRU we have illustrated the flexibility and extensibility of the framework. These protocols span a wide spectrum of complexity, from pure browsing with HTTP, through the hybrid OAI-PMH which includes selective filtering, all the way to full and comprehensive search with Z39.50. In the future, other information gathering methods could easily be added to this suite without excessive effort. An example is the Web-services based protocol used by the Fedora project; it could be described as intermediate between OAI-PMH and Z39.50 in complexity, for it provides querying ability, like Z39.50, but is borne over HTTP, like OAI-PMH.

The Exchange Center's *Download* panel provides easy-to-use interactive access to the features of these protocols. This radically lowers the complexity of information gathering from non-local information sources, which previously had to be done by invoking arcane scripts. We hope it will empower practicing digital librarians to undertake practical interoperability projects.

References

[1] D. Bainbridge, K.Y. Ke and I.H. Witten. Document level interoperability for collection creators. In *Joint ACM/IEEE Conference on Digital Libraries*, pages 105–106, Chapel Hill, 2006.

[2] David Bainbridge, Wendy Osborn, Ian Witten and David Nichols. Extending Greenstone for institutional repositories. In *Digital Libraries: Achievements, Challenges and Opportunities*, Volume 4312 of *Lecture Notes in Computer Science*, pages 303–312. Springer, 2006.

[3] Goutam Biswas and Dibyendu Paul. An evaluative study on the open source digital library softwares for institutional repository: Special reference to Dspace and Greenstone digital library. *International Journal of Library and Information Science*, Volume 2, Number 1, pages 1–010, 2010.

[4] M.V. Cundiff. An introduction to the Metadata Encoding and Transmission Standard (METS). *Library Hi Tech*, Volume 22, Number 1, pages 52–64, 2004.

[5] Richard Gartner. MODS: Metadata Object Description Schema. *JISC*, 2003. http://www.jisc.ac.uk/uploaded_documents/tsw_03-06.pdf.

[6] Dion Hoe-Lian Goh, Alton Chua, Davina Anqi Khoo, Emily Boon-Hui Khoo, Eric Bok-Tong Mak and Maple Wen-Min Ng. A checklist for evaluating open source digital library software. *Online Information Review*, Volume 30, Number 4, pages 360–379, 2006.

[7] Yan Han. Digital content management: the search for a content management system. *Library Hi Tech*, Volume 22, Number 4, pages 355–365, 2004.

[8] C. Lagoze, S. Payette, E. Shin and C. Wilper. Fedora: An architecture for complex objects and their relationships. *Journal of Digital Libraries*, Volume 6, Number 2, pages 124–138, 2006.

[9] C. Lagoze and H. Van de Sompel. The open archives initiative: building a low-barrier interoperability framework. In *Joint ACM/IEEE Conference on Digital Libraries*, pages 54–62, Roanoke, Virginia, June 2001.

[10] NISO. *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification*. National Information Standards Organization Press, 1996.

[11] M. Smith, M. Bass, G. McClella, R. Tansley, M. Barton, M. Branschofsky, D. Stuve and J.H. Walker. DSpace: An open source dynamic digital repository. *D-Lib Magazine*, Volume 9, Number 1, 2003. (doi:10.1045/january2003-smith).

[12] T. Storey. Moving Z39.50 to the web. *OCLC Newsletter*, Volume 263, 2004.

[13] D. Tidwell. *XSLT*. O'Reilly, 2001.

[14] Ian H. Witten and David Bainbridge. A retrospective look at Greenstone: lessons from the first decade. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '07, pages 147–156, New York, NY, USA, 2007. ACM.

[15] Ian H. Witten, David Bainbridge and David M. Nichols. *How to Build a Digital Library, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2009.

[16] I.H. Witten and D. Bainbridge. Creating digital library collections with Greenstone. *Library Hi Tech*, Volume 23, Number 4, pages 541–560, 2005.

[17] I.H. Witten, D. Bainbridge, R. Tansley, C.-Y. Huang and K. Don. StoneD: A bridge between Greenstone and DSpace. *D-Lib Magazine*, Volume 11, Number 9, September 2005.