Developments in Digital Preservation at the University of Illinois: The Hub and Spoke Architecture for Supporting Repository Interoperability and Emerging Preservation Standards

Thomas Habing, Janet Eke, Matthew A. Cordial, William Ingram, and Robert Manaster

Abstract

Funded by the National Digital Information Infrastructure and Preservation Program (NDIIPP), the ECHO DEPository Project supports the digital preservation efforts of the Library of Congress by contributing research and software to help society GET, SAVE, and KEEP its digital cultural heritage. Project activities include building Web archiving tools, evaluating existing repository software, developing architectures to enhance existing repositories' interoperability and preservation features, and modeling next-generation repositories for supporting long-term preservation. This article describes the development of the Hub and Spoke (HandS) Tool Suite, built to help curators of digital objects manage content in multiple repository systems while preserving valuable preservation metadata. Implementing METS and PREMIS, HandS provides a standards-based method for packaging content that allows digital objects to be moved between repositories more easily while supporting the collection of technical and provenance information crucial for long-term preservation. Related project work investigating the more fundamental semantic issues underlying the preservation of the *meaning* of digital objects over time is profiled separately in this issue (Dubin et al., 2009).

The Hub and Spoke Interoperability Architecture

HandS is a suite of tools built to support moving content between repositories while generating and maintaining PREMIS-based technical and preservation metadata. It emerged out of project activities to evaluate open-source repositories in which we found typically low out-of-the-box

LIBRARY TRENDS, Vol. 57, No. 3, Winter 2009 ("The Library of Congress National Digital Information Infrastructure and Preservation Program," edited by Patricia Cruse and Beth Sandore), pp. 556–579

support for interoperability and low support for emerging preservation standards. The next section describes the impetus and rationale behind the HandS development in more detail.

Hub and Spoke Background

The development of the Hub and Spoke (HandS) Architecture was a natural outcome of activities required to develop a test bed for evaluating multiple repository systems. During the development of our test bed we found ourselves developing a number of different though similar customized scripts and programs for exporting digital packages from one repository system and importing those digital packages into another repository system. The repository systems themselves had very little in common that would facilitate this task. They typically supported different descriptive metadata formats, had no support for provenance metadata, offered little or no support for technical metadata, and employed different means of identifying the files constituting a package. The development of an in-house tool to facilitate data interoperability between multiple repositories without the need to develop customized mechanisms for each repository combination therefore soon emerged as a key task to support our repository evaluation activities.

At the same time, we were also coming to a more structured understanding of emerging digital preservation standards, specifically early drafts of *An Audit Checklist for the Certification of Trusted Digital Repositories* (RLG, 2005; Kaczmarek et al., 2006; Kaczmarek, Habing, and Eke, 2006) and the *PREMIS Data Dictionary for Preservation Metadata* (PREMIS Working Group, 2005). We began to see that a formally developed interoperability architecture designed with a focus on providing additional support for retention of provenance and technical metadata could be a valuable and practical project deliverable, and one with immediate application in our own libraries and in other institutions that commonly implement multiple repository systems to manage and preserve digital collections.

The Need for Interoperability and Preservation Support

Institutions Commonly Rely on Multiple Repositories

There are currently many different digital repositories in widespread use, including DSpace, Greenstone, Fedora, EPrints, and CONTENTdm, along with digital archive services like those from OCLC and CDL. There are also many different sources of input into these systems, such as from Web crawlers like Heritrix or packaged content from OCLC's Web Archives Workbench, as well as numerous digitization and scanning services. It is also not uncommon for several of these systems to be in use within a single institution. If curators wish to share data internally, or with other institutions or consortia, it is very likely that multiple repository systems

will come into play. Repository interoperability issues also emerge as institutions update or replace their repository systems, and must migrate content from an existing repository system to its replacement.

Out-of-the-box Repository Interoperability Is Low

Our repository evaluation experiments and our experiences with repositories in production at our own institutions show that the native ability for repositories to interoperate is typically very basic. Almost none of the systems we tested were able to operate with one another beyond a rudimentary level, usually restricted to the OAI Protocol for Metadata Harvesting (OAI-PMH) for Dublin Core. If any OAIS concepts are implemented (and few are), such as the use of submission or dissemination information packages (SIPs and DIPs), these implementations vary greatly (Consultative Committee for Space Data Standards, 2002). In an ideal OAIS-compliant world, a DIP from one repository should be a SIP to another. However, in reality, a dissemination package produced by DSpace cannot be used for submission into EPrints. Because of these inconsistencies, achieving any real interoperability between repository systems usually entails some level of custom software development. Further, anytime a new repository is added to the mix, new software will need to be developed in order to accommodate the added repository.

Support for Emerging Preservation Standards Is Low

Few of the current repositories have any explicit support for preservation, such as for collecting preservation metadata as articulated by PREMIS, or activities to support preservation such as format migrations or checksum validations as outlined in the Trusted Digital Repository Checklist. For an institution that deploys several repository systems, a task as simple as performing consistent backups to off-line storage can become complicated by the fact that the systems store their underlying data differently. There may be data stored in relational databases, XML databases, RDF triple stores, and various file systems—all of which must be backed up, and may require different backup techniques.

Summation

The general lack of repository support for interoperability and for emerging repository standards at a time when libraries and other institutions commonly rely on multiple repository systems to manage, share, and preserve content, is the fundamental impetus behind the development of the HandS Tool Suite. The key principles of interoperability and preservation, and the approaches implemented in the HandS to support them, are examined more closely in the next section, followed by a functional and technical overview of the HandS tools.

Hub and Spoke Key Principles

The Hub and Spoke approach is based on the two key principles of interoperability and preservation, with the understanding that interoperability is not only an end unto itself, but it is also critical for preservation.

Interoperability

To reduce the complexity of interoperability, the Hub and Spoke uses a common packaging format for interchange of digital resources between different repositories. Digital packages coming from a repository are transformed into this common format before any further processing, and digital packages are transformed from this common format into the native repository format when being placed into a repository. The idea is to reduce an N^2 problem into a 2N problem as shown in figure 1.

Preservation

The second key principle is that the common packaging format as well as the processes that act on that packaging format should not only support interoperability, but should also promote preservation. This principle treats the common packaging format as an archival information package (AIP) in the OAIS model. The assumption being that one reason packages are being moved between repositories is for preservation.

There are several features of the Hub and Spoke architecture that promote preservation. One key preservation feature is the reliance on current best practices regarding preservation metadata, primarily informed

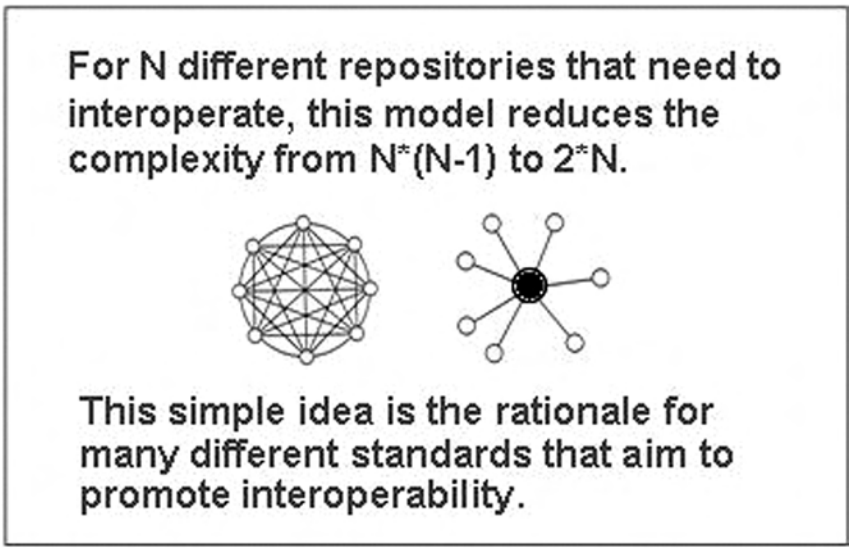

Figure 1. Interoperability Standards: A Simple Idea

by PREMIS. The Hub and Spoke is especially concerned with technical metadata about the files and bitstreams, which comprise a digital package, and also with provenance metadata about the events that occur during the package's lifetime, including events pertaining not only to the files and bitstreams, but also to the metadata itself. The technical metadata is used to validate the files and bitstreams throughout a digital object's lifetime and are updated as required, for example when a format transformation occurs. The provenance metadata is also updated throughout an object's lifetime. The tools that implement the HandS architecture perform these actions automatically as required during processing, but the data are always present in the packages so that other systems can also perform these actions as needed.

Another key preservation feature is the treatment of the packages themselves. For purposes of repository interoperability and also to support preservation, the Hub and Spoke framework treats the instantiations of the packages as first class digital objects. This means that when a HandS package is transformed for ingestion into a specific digital repository, not only are the metadata, files, and bitstreams that comprise the package decomposed as appropriate for the repository and uploaded, but the package itself (in our case a suite of METS files) is also treated as a digital object to be uploaded to the repository. Later when the digital package needs to be disseminated from the repository, not only are the metadata, files, and bitstreams available for download, but also the original HandS package. This allows the HandS system to compare the package as it was originally ingested to how it now appears as disseminated from the repository. This process, we feel, is critical for preservation in an environment of heterogeneous and changing repositories. Another aspect of this treatment of the Hub and Spoke packages as first class digital objects is that we can create snapshots of individual packages at points in time and also record preservation metadata about the package snapshots. The HandS Tool Suite currently implements this concept as a master package, which references time-stamped snapshots of the main package. The master package also records preservation metadata about the snapshots. This approach is explained in more detail in the next section, which describes the concrete implementation of the HandS packages using METS.

METS Profile

To realize the above principles, we wanted to utilize the prevailing digital library standards as much as possible. To that end, we adopted METS as the packaging standard, PREMIS as the preservation metadata standard, and MODS as the descriptive metadata standard. We also optionally utilize several format-specific technical metadata standards such as MIX and textMD for image and text objects respectively, among others. Our METS

profile, the *ECHO DEP Generic METS Profile for Preservation and Digital Repository Interoperability* (Habing, 2005), is currently registered with the Library of Congress.

As already described, the primary focus of the HandS METS profile is to enable repository interoperability and to support preservation of repository content. Because of the strong focus on preservation rather than access, the HandS profile is relatively noncommittal regarding file formats or structures; instead, special attention is given to administrative and technical metadata, particularly to integrating the PREMIS data model and schema into METS. We anticipate that our file format-agnostic HandS profile may be overlaid on top of, or inherited by, other profiles that better define a particular file format or structure, providing them with added support for preservation or interoperability. An example of this arrangement, where a format-specific METS profile is implemented as a subclass of the PREMIS-focused HandS profile, is the *ECHO DEP METS Profile for Web Site Captures* (Habing, 2006), also registered with the Library of Congress. Thus the *ECHO DEP Generic METS Profile for Preservation and Digital Repository Interoperability* is generally not concerned with rendering or making accessible any particular representation of an object, but it is concerned with preserving the object and its representations, including the history of how those have changed over periods of time. In this context, preservation refers to short-term interoperability, preserving the representations and metadata as a digital package is moved between two different repositories. It also refers to the long-term preservation of the package and its history as it exists in various repositories for long periods of time and undergoes various "preservation actions" such as fixity checks, normalizations, or format migrations.

Note that though the profile is generally agnostic about almost all aspects of a digital object's representation, such as structure or file formats, we have made some pragmatic concessions, such as mandating at least MODS for the primary descriptive metadata (dmdSec) while at the same time allowing multiple alternative descriptive metadata sections. The alternative descriptive metadata sections are used as a means to record various versions of these metadata as they have existed in different repositories or at different points in time. A potential usage scenario can be illustrated in the following migration example using our METS profile:

- We start with a digital object whose original source descriptive metadata is in the MARCXML format. Because our profile requires MODS as the primary descriptive metadata, the MARCXML will be transformed into MODS, and the MODS will be stored in the METS document along with a provenance statement with some details about the transformation, especially identifying the source metadata format. However, because descriptive metadata are considered to be a significant part of the rep-

resentation of an entity and because transformations between metadata formats are often imperfect, the original MARCXML format is also stored in the METS document as an alternate metadata format.

- Now suppose that the digital object is to be ingested into DSpace. DSpace, however, does not have native support for the MODS or MARCXML metadata formats; therefore, as part of the ingest process, the MODS must be transformed into the idiosyncratic Dublin Core (DC) metadata format that is supported by DSpace. This metadata format is also added as another alternate descriptive metadata format to the METS document, along with a provenance statement describing how this new DC format was derived from the primary MODS format.
- Next imagine that the object exists in DSpace for some period of time during which the descriptive metadata undergoes some revision, such as the addition of new subject terms or the addition of an abstract. Now the object is to be disseminated from DSpace for ingest into some new repository. This could trigger the addition of another alternate descriptive metadata section to the METS document. This alternate format would conform to the idiosyncratic DSpace Dublin Core format, but the provenance statement would specify that this DC format represents a newer version of the descriptive metadata than was originally ingested into DSpace.

The above scenario would produce a chain of descriptive metadata formats, such as MARCXML (original) → MODS (primary) → DC (version 1) → DC (version 2), with provenance PREMIS event statements adequate to determine the sequence of events that led to this chain. As part of this profile we also envision future processes that might reconcile later metadata revisions and merge those revisions back into a new primary MODS descriptive metadata section. The preservation of semantics during these types of migrations is one of the concerns of semantic preservation described in Dubin et al. in this issue.

Because we feel that administrative metadata are important for preservation, this profile is fairly prescriptive when it comes to the administrative metadata, which can be associated with almost all of the sections that make up a representation: structures, files, and bitstreams, and descriptive metadata. Particular attention is paid to the technical and provenance metadata associated with these METS sections.

Master METS Profile

Another key idea behind our METS profile is the idea of a Master METS document. Each package in the HandS architecture consists of a single Master METS document, one or more METS Snapshot documents, plus all the files and bitstreams that are referenced from the METS Snapshots, as shown in figure 2.

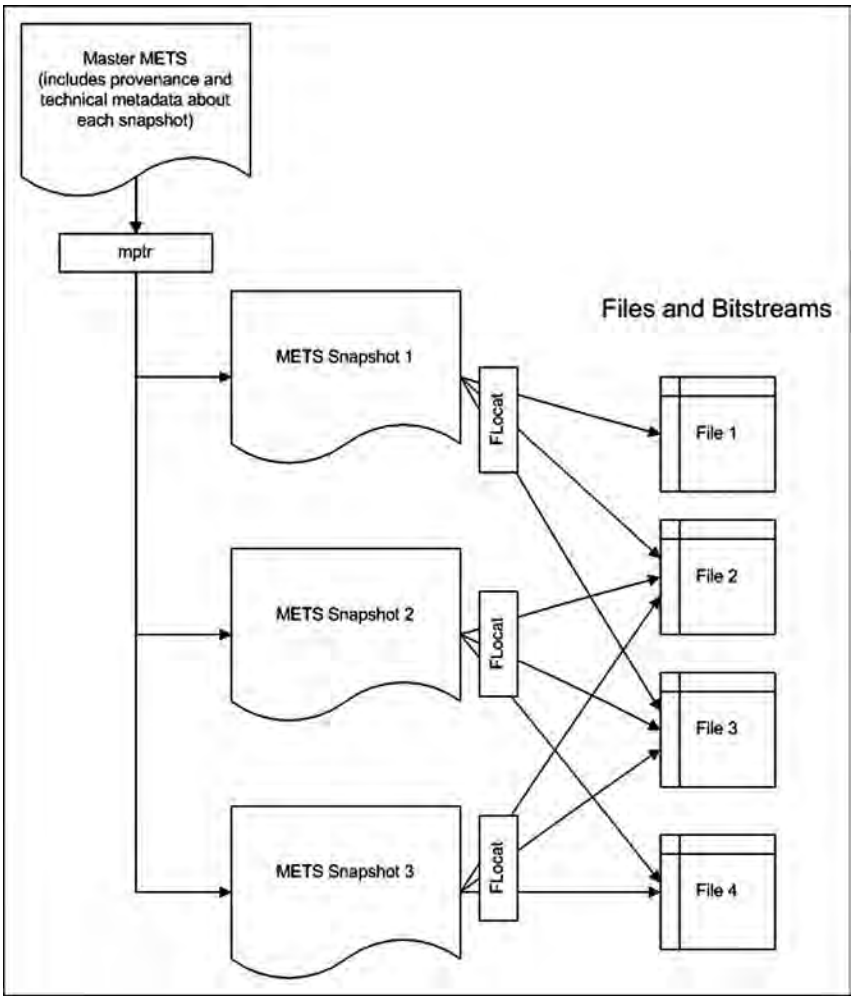

Figure 2. Master METS Showing Multiple Snapshots and Associated Files

Each Snapshot represents a version of the digital package at a point in time, usually when the package is either retrieved from or placed into a given repository. Nearly any aspect of a digital object's representation may change with time, including descriptive metadata, structure, and, as illustrated above, even the files referenced from a package, perhaps as format migrations occur. These changes are recorded as provenance statements in the METS Snapshot in which the change is manifest. For example, METS Snapshot 2 in the above diagram would have a PREMIS event describing that File 1 was deleted from the package and File 4 was added

to the package. In most cases, the HandS system can automatically detect when these changes occur and will automatically add the appropriate provenance statements or embellish the technical metadata as required. However, it may not be able to determine why the changes occurred without some sort of intelligent intervention. HandS is able to detect the changes because it has access to the previous Snapshots and can compare the Snapshot of the package as it went into a repository to the package that is retrieved from the repository. This is one of the primary reasons that the METS documents themselves are also placed into a repository along with the other files that are actually part of the package.

Summation

The METS profile implementations described above are an integral piece of the HandS architecture, used as framework for generating and maintaining PREMIS-based metadata over time to support long-term preservation. The next section looks in more detail at other mechanisms of the HandS Tool Suite, and illustrates its overall workflow cycle.

HandS Workflow Cycle

As described in the preceding sections, the HandS Tool Suite provides a framework for sustaining and enriching preservation metadata for digital objects as they are moved into, out of, and between digital repository systems. Digital objects or preservation packages typically refer to a set of files that represents a single intellectual entity, including metadata about the entity or about the files themselves. In the Hub and Spoke workflow cycle (see fig. 3), digital objects are retrieved, converted to a common profile, validated, enriched with metadata, transformed into a repository-compatible form, and ingested into a digital repository.

Workflow Overview: GET, PROCESS, PUT

Preservation packages may enter the HandS workflow in various ways: some may come from third-party applications like the OCLC Web Archives Workbench, others may be disseminated from a digital repository like DSpace or EPrints, and some will originate simply as directories of files on a computer file system. In any case, the set of files that make up the preservation package must first be gathered and organized for processing. Objects entering the workflow from a digital repository system must first be fetched from the repository by interacting with its native dissemination routine, which will vary from repository to repository. This interaction with the repository system is facilitated by our Lightweight Repository Create, Retrieve, Update, and Delete Service—affectionately named LRCRUD. LRCRUD is made up of two modules: the LRCRUD Client, which runs on the same machine as the other HandS tools, and the LRCRUD Service, which runs alongside a digital repository system. To re-

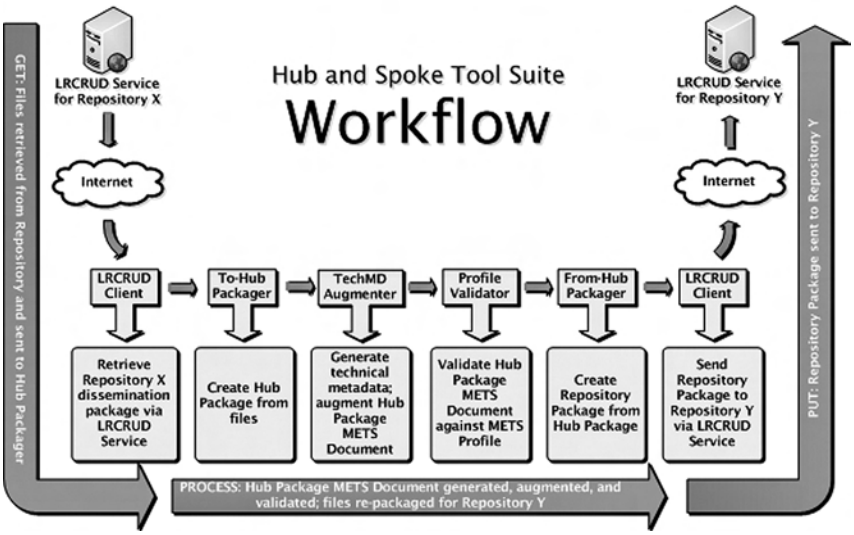

Figure 3. HandS Workflow Cycle

trieve a package from the repository, the LRCRUD Client makes a request to the LRCRUD Service. The LRCRUD Service, in turn, communicates directly with the repository system and retrieves the package via the repository's native dissemination routine. The LRCRUD Service zips up the package and sends it over the network to the Client. Once the package has been received by the LRCRUD Client and verified, its contents are unzipped onto the local file system.

From there, the To-Hub Packager tool converts the digital object into what we call a Hub Package. A Hub Package is made up of the content files that constitute the digital object; METS documents containing descriptive, administrative, and structural metadata about the object at various points in time; and a single Master METS document that comprises chronological and structural information about the other METS documents. The Master METS file will contain a pointer to at least one, but potentially several other ECHO DEP METS documents, each of which serves as a snapshot of the Hub Package at some point in its lifecycle. The ECHO DEP METS document is the heart of a Hub Package; it holds together all the files and various metadata that make up the package. When a Hub Package is created, a new ECHO DEP METS document is generated for the package. If the package already contains an older ECHO DEP METS document (generated prior to ingestion into the repository), the new METS document is compared to the older one to discover any changes or

damages to the package that might have occurred while in the custody of the repository.

The ECHO DEP METS document is then enriched with technical metadata and validated against the ECHO DEP METS Profile registered with the Library of Congress (Habing, 2005). The TechMD Augmentor tool enriches the METS document with format-specific technical metadata found by analyzing each of the package's content files, and converting the result into PREMIS Object metadata. Once the package has been analyzed and enriched, the Profile Validator closely inspects the constituent files that make up the package, both data and metadata, and verifies there are no errors or inconsistencies.

At this point, the Hub Package is ready to be sent on to another repository. But first, it has to be converted into a form compatible with ingestion into the target repository, which again will vary from repository to repository. This final conversion is carried out by a From-Hub Packager tool built specifically for the target repository. From there, the package is handed off from the LRCRUD Client to the LRCRUD Service for the target repository and ingested.

The workflow cycle might be more easily understood by following an example preservation package as it makes its way through the process. For this example, we will use three small files that make up a single Web page: an HTML file, a Cascading Style Sheet (CSS), and a JPEG image. These three files compose a single preservation package, or item, which has been submitted to a DSpace digital repository. Using the HandS Tool Suite, we will transfer the item from DSpace to an EPrints repository, while generating preservation and technical metadata along the way.

Step 1: Retrieve Repository X Dissemination Package via LRCRUD

In our example, suppose we have an LRCRUD Service running alongside a DSpace repository on a remote server. The LRCRUD Client application sends a request to the LRCRUD Service to retrieve (GET) a package from the repository. The LRCRUD Service relays the retrieval request to DSpace using the repository's native dissemination method. The output of a repository's dissemination will typically be made up of any number of metadata streams and other supporting artifacts in addition to the item's content files. In the case of DSpace, the package will include a DSpace METS file that encompasses MODS descriptive metadata about the package and PREMIS technical metadata pertaining to each of the constituent bitstreams. In our example, the package returned by DSpace now contains four files: the HTML, CSS, and JPEG files we began with, and a DSpace METS file.

The LRCRUD Service receives the DSpace dissemination and packages its contents into a zip archive, which will be transmitted over HTTP to the LRCRUD Client. The LRCRUD Service also calculates file size and a checksum value for the zip file before sending it, and transmits these values as Content-MD5 and Content-Length HTTP header fields along

with the package zip file. As the LRCRUD Client receives the package zip file, it too calculates file size and checksum values, which are validated against the HTTP header fields to ensure the package was unharmed during the file transfer. Assuming the values agree, the package is unzipped and saved to disk.

Step 2: Create Hub Package from Repository Dissemination Files

To create a Hub Package from the repository dissemination package, the To-Hub packager needs to produce a new ECHO DEP METS document for the package. The packager begins by searching the retrieved files for any metadata included by the repository. In our example, the packager locates the DSpace METS document and retrieves its MODS descriptive metadata stream. This DSpace MODS metadata will be transformed into Aquifer MODS and inserted into the new ECHO DEP METS document's descriptive metadata section. Other repositories export metadata in different formats (e.g., Dublin Core), but in all cases the package metadata is ultimately transformed to Aquifer MODS by the To-Hub Packager.

The To-Hub Packager then creates an entry in the file section of the ECHO DEP METS document for each of the package's constituent files. In our example, the new ECHO DEP METS document will contain a file element for each of our three content files. The To-Hub Packager will also create in the new ECHO DEP METS document three PREMIS technical metadata objects to correspond to three file elements. Each will contain basic technical metadata about one of the files, including checksum values, file size, and MIME-type. Finally, any leftover descriptive and technical metadata elements from the DSpace METS document are inserted into the ECHO DEP METS document as alternate metadata so that it is never lost.

If the package contains older ECHO DEP METS documents (because it had been packaged by HandS before entering the repository), the most recent ECHO DEP METS document is compared to the just-generated ECHO DEP METS document to expose any changes the package may have undergone since it was last analyzed. These data are recorded in the new ECHO DEP METS document's provenance metadata as PREMIS events. If the package contains a Master METS document, a pointer to the new METS document is created and designated as the most-current ECHO DEP METS document for the package. If no Master METS document can be found, the To-Hub Packager creates one from scratch. Once the new METS document has been created and the Master METS document is updated, Hub Package creation is complete. Our example Hub Package now consists of a Master METS document, which points to a single ECHO DEP METS document. This ECHO DEP METS document contains descriptive metadata about the package; technical metadata about each of the three content files, along with pointers to those files and the technical metadata left by DSpace; and provenance metadata documenting the package's export from the repository and its Hub Package transformation.

Step 3: Generate Technical Metadata; Augment Hub Package METS Document
Using tools from the JSTOR/Harvard Object Validation Environment (JHOVE), the HandS TechMD Augmenter module analyzes each of the Hub Package's content files and generates format-specific, technical metadata for each. The JHOVE-generated metadata is transformed using format-specific XSLT stylesheets, and inserted into the technical metadata section of the ECHO DEP METS document. Any inconsistencies between the technical metadata currently held in the METS document and metadata generated by JHOVE are recorded in the provenance section of the ECHO DEP METS document as PREMIS validation events. The technical metadata stored in the ECHO DEP METS document is formatted in compliance with the following metadata preservation standards: AudioMD for audio files; TextMD for text, XML, and HTML; and MIX for images.

In our example the HTML, CSS, and JPEG files will each be analyzed by JHOVE. The JHOVE output for both the HTML and CSS files will be formatted as TextMD, and the output for the JPEG image will be formatted as MIX. Each will be inserted into the ECHO DEP METS document in a technical metadata element corresponding to the appropriate file element. The JHOVE analysis itself is also documented and recorded in the ECHO DEP METS document as a validation event. One of the limitations of JHOVE is its small number of supported media types. In its current release JHOVE offers no support for closed formats such as Microsoft Office files. Another drawback of using JHOVE is that it only reports the MIME-type correctly for HTML or XML files if they are well formed; otherwise it reports them as plain text, causing discrepancies within the ECHO DEP METS document and validation warnings. Nevertheless, we found JHOVE to be a useful tool for analyzing files and generating technical metadata. For more on JHOVE visit http://hul.harvard.edu/jhove/.

Step 4: Validate Hub Package METS Document against METS Profile
The Profile Validator examines the current ECHO DEP METS document for the Hub Package against the requirements of our METS profiles currently registered with the Library of Congress (Habing 2005, 2006).

Key validation points include checking to make sure that the primary descriptive metadata element contains a MODS object that conforms to the Aquifer MODS profile; that every file referenced by the file section has associated technical metadata PREMIS objects; and that all provenance metadata associated with a file contain valid PREMIS event elements (DLF Aquifer Metadata working Group, 2006). The Profile Validator also checks that the package content files referenced by the ECHO DEP METS document are accounted for, and that their checksum, file-size, and mime-type values are correct.

Our example ECHO DEP METS document passes validation for the following reasons: it contains valid Aquifer MODS in its primary descriptive metadata element; each of its file elements reference technical meta-

data elements containing valid and complete PREMIS object metadata; and it conforms structurally to our METS profile requirements. Once the validation has completed, the validation event itself is documented and recorded in the ECHO DEP METS as a PREMIS validation event.

Step 5: Create Repository Package from Hub Package
Before a repository can accept a package for submission, it must first receive a description of the package's contents. The From-Hub Packager module uses descriptive metadata extracted from the Hub Package ECHO DEP METS document to generate the repository-specific metadata need for package submission. This process usually involves transforming the Aquifer MODS metadata found in the ECHO DEP METS document into a metadata format required for repository submission, and will vary from repository to repository.

In our example, we are sending the package to an EPrints repository, which means the packager will generate an EPrints-specific metadata file from the Aquifer MODS stream. The transformation event is recorded in the ECHO DEP METS document as a PREMIS metadata-transformation event, and the newly-generated metadata is added to the METS document as alternate descriptive metadata. A Repository Package zip file is then created, consisting of the Master METS document, all the subordinate METS snapshot documents and the content data files, as well as any repository-specific metadata files.

Step 6: Send Repository Package to Repository Y via LRCRUD
At this last step in our example, we have an EPrints-specific LRCURD Service running on a remote server with an EPrints repository. The LRCRUD Client sends a request to the LRCRUD Service to create (POST) a new package. The LRCRUD Service relays the create request to the repository and, using the repository's native methods, creates an empty record. The LRCRUD Service receives a new location identifier, or handle, corresponding to the newly created location in the repository, which it sends back to the LRCRUD Client. This location identifier is inserted into the package's ECHO DEP METS document as the primary ID for the METS document.

The LRCRUD Client then sends a request to the LRCRUD Service to update (PUT) the new package at that location. The LRCRUD Client calculates file size and checksum values for the package zip file before sending it to the Service, and it transmits these values as Content-MD5 and Content-Length HTTP header fields along with the package. As the LRCRUD Service receives the package zip file from the Client, it calculates its own file size and checksum values and validates them against the HTTP header fields to ensure the package was unharmed during the file transfer. Once the LRCRUD Service has validated the file transfer, it unzips the package and ingests each of its contents—including the package METS files—into the repository using the repository's native submission routine. The repository-specific descriptive metadata that was generated

in Step 5 above is submitted to the repository as well. Once the package has been fully ingested, the LRCRUD Service returns an update response message to the LRCRUD Client confirming the successful submission, or an error if the submission failed.

Some repositories allow for certain bitstreams to be given privileged status. In such cases the Master METS and ECHO DEP METS files may receive special status; but in all cases the METS files are preserved along with the other package content files and are treated as first class objects with regard to the repository. That way, when the package is retrieved from the repository, all the metadata pertaining to the state of the package before it was submitted to the repository is not lost.

Summation

Through the workflow process described above, HandS provides tools to facilitate moving digital objects between multiple repositories while generating and maintaining important PREMIS-based technical and provenance preservation metadata. Digital objects are retrieved, converted to a common profile, validated, enriched with metadata, transformed into a repository-compatible form, and ingested into the target repository.

HandS Technical Implementation

The key technical components of the HandS implementation are the Hub and Spoke METS profile Java classes, providing an extensible Java API to our METS profile with Apache XMLBeans; the To- and From-Hub Packager modules, facilitating interoperability through pluggable interfaces; and the Lightweight Repository CRUD Service (LRCRUD), supporting the dissemination and submission of objects by defining a protocol for transmitting digital objects to and from repository systems over HTTP.

Hub and Spoke METS Profile API

The core of the HandS Tool Suite is our METS Profile API, a Java code representation of a METS XML document compiled from our METS profile. The bulk of our METS classes were created with Apache XMLBeans (http://xmlbeans.apache.org/), a tool for generating Java classes from XML schema files (XSD files). With XMLBeans, we are able to compile XML schema documents to produce a Java code structure, allowing us to work with XML data through our own Java classes and methods. To create our METS profile API, we combine methods from XMLBeans-generated classes from the METS, MODS, and PREMIS schemas, along with format-specific preservation metadata schemas like MIX, TextMD, and AudioMD. We also layer custom-built convenience methods on top of the XMLBeans-generated methods to facilitate additional manipulation of the METS document in a fashion unique to our METS profile.

A new HandS Profile Java object can be created from scratch given a set of content files and accompanying metadata, or by instantiating an

existing XML METS document that conforms to our profile. Once instantiated, the underlying METS document object can be operated upon programmatically through API calls. In working with the API, we are assured that any manipulation of the METS document will always be consistent with our METS profile. For instance, to add a new file to the preservation package, a call is made to the addFile() method, which in turn triggers calls to other methods that ensure the METS object remains consistent with our profile—such as adding a new PREMIS Object techMD section associated with the new file, and generating checksum, MIME-type, and file size values. At any time the METS object can be validated against our profile, or re-serialized as XML and saved to the file system.

To- and From-Hub Packagers

To facilitate repository interoperability, the HandS Tool Suite includes a set of packager classes for transforming a collection of preservation items into a Hub Package, and for transforming a Hub Package into a form required for submission into a given repository. For items entering the Hub and Spoke from a digital repository, the repository-specific To-Hub packager takes the native repository dissemination, unpacks it, and instantiates a new HandS METS Profile object from its contents. Going the other way, a repository-specific From-Hub packager prepares a Hub Package for submission into the particular repository.

Currently, we have To-Hub packagers for processing items coming from DSpace, EPrints, OCLC's Web Archives Workbench, or from a directory of files. Our current list of From-Hub packager targets includes DSpace, EPrints, and the Library of Congress archive standard Bagit. To- and From-Hub packager modules for the Fedora repository are currently in development. We have employed a "pluggable" architecture for creating packager modules. Base To- and From-Hub classes are implemented in Java as abstract classes with the intention that they will be overridden and extended by other programmers needing to tailor the HandS Tools to their specific repository or archiving standard. This modular architecture allows other developers to create packager plug-ins for their own repository systems without having to recompile or re-factor the existing HandS codebase.

Lightweight Repository CRUD Service (LRCRUD)

The Lightweight Repository CRUD service specification defines dissemination and submission Web-service interfaces to digital repository systems for use with the ECHO DEP Hub and Spoke Tool Suite. The LRCRUD specification defines a protocol for transmitting digital objects to and from repository systems over HTTP. It enables users to obtain objects in a format expected by the HandS processing scripts and supplies digital objects to repositories in a format expected by their native ingestion mechanisms. The specification is implementation-agnostic: it simply defines the

parameters and responses required to enable a service implementation to communicate with the LRCRUD client application. This allows LRCRUD implementers to choose the most appropriate environment and programming languages for interacting with their chosen repository. The HandS Tool Suite currently has LRCRUD implementations for DSpace, EPrints, and Fedora.

The LRCRUD Service follows Representational State Transfer (REST) conventions. It exposes CRUD actions on repository content over the HTTP protocol. As mentioned above, CRUD is an acronym for Create Retrieve Update and Delete—the basic operations that applications should implement when acting upon persistent storage like relational database management systems, file systems, and the like. The LRCRUD client communicates with the LRCRUD service via HTTP methods, status codes, and headers. The list below shows how the CRUD actions are mapped to the HTTP methods:

- Create == POST
- Retrieve == GET
- Update == PUT
- Delete == DELETE

In most cases the LRCRUD service will reside on the same host as the repository it serves so that it has access to the repository's API.

LRCRUD is essentially a "dumb" packager; it is simply a way to supply files to the remote repository in any format/configuration that it can natively ingest. In this it is similar to protocols like the Simple Web Service Offering Repository Deposit, or SWORD (Allinson, François, and Lewis, 2008; JISC, 2008), which are being adopted by repositories—and which may make the submission function of LRCRUD ultimately unnecessary.

It may be beneficial to present some descriptive step-by-step examples in order to clarify the functions of the LRCRUD components within the Hub and Spoke Tool Suite. These examples describe in detail the interactions between the client and the service.

LRCRUD Functions—Examples

Dissemination

Dissemination (see fig. 4) is the act of retrieving an item from a repository, whereby the item is defined as an intellectual entity comprising any number of content streams, metadata streams, and other supporting artifacts. Items disseminated from a repository using the LRCRUD service are most likely bound for processing and transformation by the HandS Tool Suite To-Hub packager. The packager creates METS files conformant to the HandS profile, extracts and augments technical metadata, and records provenance information.

Described below are the four major steps in negotiating dissemination:

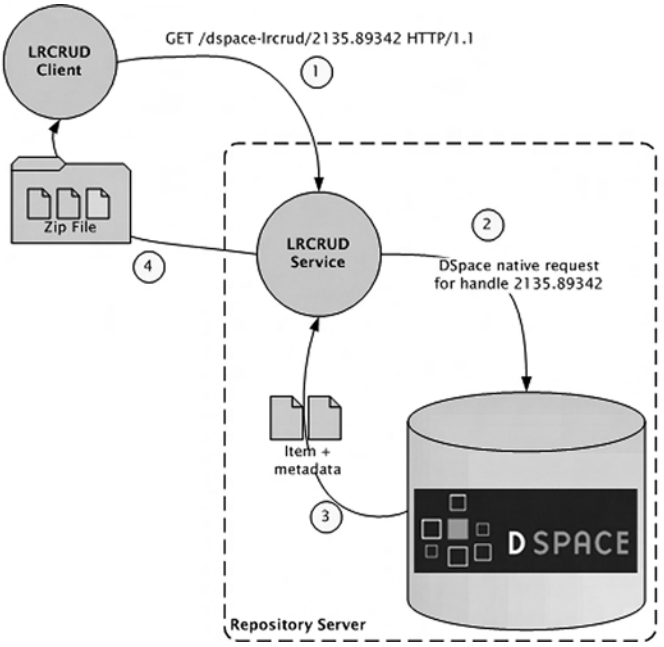

Figure 4. LRCRUD Dissemination

- The LRCRUD client submits an HTTP GET request to the LRCRUD service. The GET request provides the ID of the item desired via the LRCRUD service URL syntax.
- The service calls the repository's native dissemination routine for the ID indicated.
- The service receives the output from the dissemination and adds the entire content into a zip file.
- The service returns the zip file containing the "package" to the client.

Submission

Submission (see fig. 5) is the act of either (1) adding an item to a repository for the first time or (2) updating an item already in the repository. Described below is the process of adding an item to a repository for the first time. This is a two-stage process; the first stage reserves an identifier in the system, while the second actually places content in the repository.

Stage 1: Create stub record to reserve an identifier. It is critical to note that the package itself is not uploaded as part of the POST request; rather, the POST request creates only a stub or placeholder record. The reason that the actual package is not uploaded as part of the POST is that the identifier assigned to the package by the repository needs to

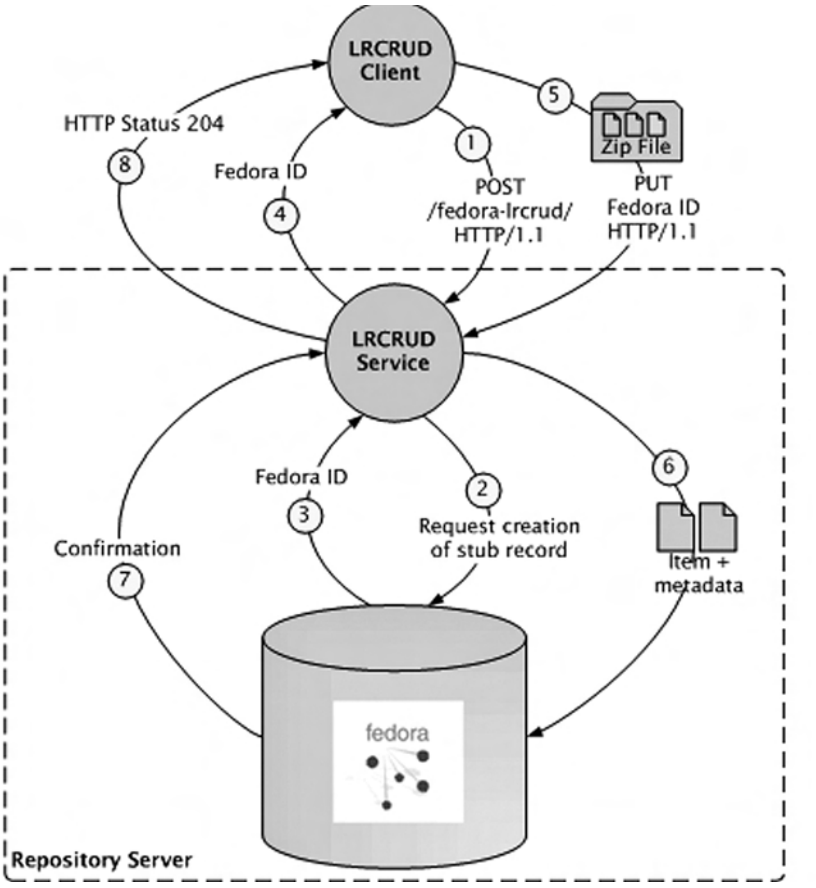

Figure 5. LRCRUD Submission

be embedded in the METS file, which is part of the package. The typical sequence of operations to ingest a new package is to use POST to create a new placeholder record and get the identifier for that record. That identifier is then used to update provenance and other metadata that is part of the package, and then the placeholder record is updated or overwritten with the actual package using the PUT action.

The major steps in this process are:

- the LRCRUD client issues a POST request to the LRCRUD service specifying the ID of "where" to create the record (e.g., in a specific collection) if needed;
- the service calls the repository's native item or ID creation routine;
- the repository supplies the service with the ID for the newly created record;

- the service responds to the client with an HTTP 201 "Created" message and returns the ID in the Location: header.

Stage 2: Upload and ingest the item. In this stage, the item is uploaded and placed in the repository. This is the exact process for updating an existing item:

- the LRCRUD client issues a PUT request to the LRCRUD service to replace the package identified by the supplied URI. The entity body of the request will contain a zip file containing the "package" to be ingested;
- the service unpacks the files and calls the repository's native ingestion routine;
- the service responds to the client with an HTTP 204 "No Content" message indicating that the request was successful.

Lessons Learned

Below, in no particular order, are several key lessons learned during the course of developing the HandS architecture.

Merging Metadata Is Potentially Risky

After expending much effort exploring how we might merge different versions of metadata files so as to maintain a single master metadata file, we reached the conclusion that this was a very difficult problem, and potentially dangerous in terms of data loss. This realization led us to our current architecture that skirts the issue of merging metadata into a single file by maintaining multiple Snapshot METS files all referenced from a common Master METS file.

METS Supports Multiple Metadata Formats Well

Combining PREMIS, MODS, and other XML-based technical metadata formats into a single METS document worked well for this particular project. The general structure of METS seemed to lend itself to constructing preservation packages. Our conceptual model, which was directly influenced by the METS and PREMIS structures, consisted at a high level of the intellectual entity having one or more representations. These representations and all their component parts were the primary focus of the preservation efforts. The METS file itself is treated as the abstract parent representation of the intellectual entity. However, there are also one or more concrete representations consisting of each structMap within the METS file. These representations consist of the relationships embodied in the structMap (and possibly the related structLink sections); the files and bitstreams referenced from the structMap; and the associated descriptive metadata (dmdSec), which could be referenced via the structMap or via individual files or bitstreams. All remaining parts of the METS document, primarily the header (metsHdr) and administrative metadata (amdSec) sections, are not considered part of the intellectual entity's representa-

tions but are, instead, *metadata about* these representations—mostly concerned with preservation and thus having a strong focus on technical and provenance metadata. There were pragmatic challenges in getting these disparate metadata standards to work together, however, and the next paragraph conveys one such example.

Implementing PREMIS in METS Requires High Level Structural Decisions
Embedding PREMIS metadata within a METS package was not an intuitive undertaking. There were several reasons for this. Among these were various overlaps in the metadata fields supported by each standard. When faced with these overlaps our general approach was to provide the metadata in both places. Although this approach introduced duplication and the opportunity for inconsistencies into the metadata, we feel the added flexibility in processing compensated for these shortcomings. Moreover, the HandS Tool Suite validation steps ensured that these types of inconsistencies were not present. Decisions were also required as to where to embed the PREMIS entities within the METS file. While these entities are clearly all administrative metadata, they do not always fit neatly within one of the four subgroups—techMD, digiprovMD, sourceMD, or rightsMD—provided by METS. Refer to the ECHO DEP METS profiles (Habing, 2005) for details. Project staff also participated in a working group chaired by Rebecca Guenther at the Library of Congress to address this issue. The working group produced a report, *Guidelines for using PREMIS with METS for Exchange* (Guenther, 2008).

Next Steps: The Hub and Spoke

Development of the Hub and Spoke Tool Suite is ongoing. The latest versions of the source code can be downloaded from the project's SourceForge website at http://sourceforge.net/. Recent developments include the addition of a From-Spoke for the Bagit specification (Boyko et al., 2008) and modifications to support version 1.5 of DSpace. Work is continuing apace on both From- and To-Spokes for the Fedora repository with particular attention being paid to how our METS profile might be accurately mapped to a Fedora content model, reducing the need for potentially lossy mappings as have thus far been required for other repository systems. The project is also looking at other potential repositories, such as LOCKSS or CONTENTdm, for Spoke development. In addition to developing new Spokes, we are also monitoring developments with the next version of JHOVE, as well as with the Global Digital Format Registry (GDFR, n.d.), to explore how these tools might be used to enhance the format-specific technical metadata we are currently generating for different file types.

Supporting Preservation Now and in the Future
The Hub and Spoke (HandS) framework enhances the interoperability and preservation features of existing open-source repository systems. It

provides a suite of tools to facilitate moving digital objects between repositories more easily while supporting the collection of technical and provenance information crucial for long-term preservation. It is intended to support curators' efforts today to manage content in multiple repository systems and to preserve valuable preservation data in accordance with emerging digital preservation standards.

In the long term, however, we see the need for the next generation of digital repositories to do more in order to support our ability to preserve the *meaning* of the digital objects maintained in repositories. Current repository systems preserve the *structures* of digital objects, from which meaning or semantics must be inferred. Learning from real-world data migration examples from the HandS efforts, GSLIS and NCSA researchers are working to model how semantic inference capability may help next-generation archives preserve the meaning (not just the structures) of digital objects and head off longer-term preservation risks. Specifically, we are developing automated reasoning techniques targeted at identifying, and eventually correcting, problematic metadata descriptions. This work is profiled separately in this issue by Dubin et al.

Conclusion

With digital preservation still in its infancy, many changes to emerging standards, strategies, and methodologies can be expected in the coming years. The Hub and Spoke framework provides a model that attempts to incorporate current technologies and best practices from the field to support digital preservation in current repository environments. It implements METS and PREMIS to provide a standards-based method for packaging content that allows digital objects to be moved between repositories more easily while supporting the collection of technical and provenance information crucial for long-term preservation. HandS is intended to help curators of digital objects today by providing improved support for preservation and interoperability to existing repository systems. Ultimately, in order to meaningfully preserve our digital content over time, we will need the next generation of preservation tools to support automatic inference of meaning, or semantics, from changed—and thus potentially ambiguous—information structures.

Acknowledgments

The authors would like to acknowledge the generous support from the Library of Congress, through the National Digital Information Infrastructure and Preservation Program, which made possible the research and development of the preservation and interoperability technologies that are the focus of this article.

References

Allinson, F., François, S., & Lewis, S. (2008, January). SWORD: Simple Web-service offering repository deposit. *Ariadne, 54*. Retrieved September 11, 2008, from http://www.ariadne.ac.uk/issue54/allinson-et-al/

The Apache Software Foundation. (2008). *Welcome to XMLBeans*. Retrieved September 11, 2008, from http://xmlbeans.apache.org/

Boyko, A., Kunze, J., Littman, J., & Madden, L. (2008). The Bagit file package format (V0.95). Retrieved September 15, 2008, from http://www.cdlib.org/inside/diglib/bagit/bagitspec.html

Consultative Committee for Space Data Standards. (2002). *Reference model for an Open Archival Information System (OAIS)*. CCSDS 650.0-B-1. Blue Book. Retrieved September 12, 2008, from http://public.ccsds.org/publications/archive/650x0b1.pdf

DLF Aquifer Metadata Working Group. (2006). *Digital Library Federation / Aquifer Implementation Guidelines for Shareable MODS Records*. Retrieved September 10, 2008, from http://wiki.dlib.indiana.edu/confluence/download/attachments/24288/DLFMODS_ImplementationGuidelines_Version1-2.pdf?version=1

Dubin, D., Futrelle, J., Plutchak, J., & Eke, J. (2009). Preserving Meaning, Not Just Objects: Semantics and Digital Preservation. *Library Trends*, 57(3), 595-609.

The ECHO DEPository project. (n.d.). Retrieved September 9, 2008, from http://ndiipp.uiuc.edu/.

Global Digital Format Registry (GDFR) Information Site. (n.d.). Retrieved September 15, 2008, from http://www.gdfr.info/

Guenther, R. (2008). *Guidelines for using PREMIS with METS for exchange*. Retrieved September 11, 2008, from http://www.loc.gov/standards/premis/guidelines-premismets.pdf

Guenther, R. (2008). Battle of the buzzwords: Flexibility vs. interoperability when implementing PREMIS in METS. *D-Lib Magazine, 14*(7/8). Retrieved September 12, 2008, from http://www.dlib.org/dlib/july08/guenther/07guenther.html

Habing, T. G. (2005). *ECHO Dep generic METS profile for preservation and digital repository interoperability*. Retrieved September 10, 2008, from http://www.loc.gov/standards/mets/profiles/00000015.xm;

Habing, T. G. (2006). *ECHO Dep METS profile for Web site captures*. Retrieved September 10, 2008, from http://www.loc.gov/standards/mets/profiles/00000016.xml

Habing, T. G. (2007). *Lightweight repository CRUD Service (LRCRUDS)*. Retrieved September 10, 2008, from http://dli.grainger.uiuc.edu/echodep/hands/LRCRUDS.htm

JISC. (2008). SWORD. Retrieved September 11, 2008, from http://www.ukoln.ac.uk/repositories/digirep/index/SWORD

JHOVE: JSTOR/Harvard Object Validation Environment. (2007). Retrieved September 11, 2008, from http://hul.harvard.edu/jhove/

Kaczmarek, J., Habing, T. G., & Eke, J. (2006). Repository software evaluation using the audit checklist for certification of trusted digital repositories. In *Proceedings of the 6th ACM/IEEE-CS joint conference on digital libraries 2006, Chapel Hill, NC, USA June 11-15, 2006*. New York: Association for Computing Machinery. Retrieved September 10, 2008, from http://doi.acm.org/10.1145/1141753.1141774

Kaczmarek, J., Hswe, P., Eke, J., & Habing, T. G. (2006). Using the "Audit checklist for the certification of a trusted digital repository" as a framework for evaluating repository software applications. *D-Lib Magazine, 12*(12). Retrieved September 10, 2008, from http://www.dlib.org/dlib/december06/kaczmarek/12kaczmarek.html.

METS: Metadata encoding & transmission standard, official website (2008). Retrieved September 10, 2008, from http://www.loc.gov/standards/mets/

MIX: NISO metadata for images in XML schema, technical metadata for digital still images standard, official website. (2008). Retrieved September 10, 2008, from http://www.loc.gov/standards/mix/

MODS: Metadata object description schema, official website. (2008). Retrieved September 10, 2008, from http://www.loc.gov/standards/mods/

OAI-PMH: Open Archives Initiative—Protocol for Metadata Harvesting. *(2008)*. Retrieved September 12, 2008, from http://www.openarchives.org/pmh/

PREMIS: Preservation metadata maintenance activity, official website. (2008). Retrieved September 10, 2008, from http://www.loc.gov/standards/premis/

PREMIS Working Group. (2005). *Data dictionary for preservation metadata*. Dublin, OH: OCLC and RLG. Retrieved September 10, 2005, from http://www.oclc.org/research/projects/pmwg/premis-final.pdf

RLG. (2005). *An audit checklist for the certification of trusted digital repositories.* Mountain View, CA: RLG. Retrieved September 10, 2008, from http://worldcat.org/arcviewer/1/OCC/2007/08/08/0000070511/viewer/file2416.pdf

textMD: Technical Metadata for Text, official website. (2008). Retrieved September 10, 2008, from http://www.loc.gov/standards/textMD/

UIUC Echodep Hub and Spoke Framework Tool Suite. (n.d.). Retrieved September 12, 2008, from http://dli.grainger.uiuc.edu/echodep/hands/

Thomas Habing is a research programmer at the Grainger Engineering Library Information Center at the University of Illinois at Urbana-Champaign where for the past ten years he has worked on various digital library projects. Tom currently spends half his time as a developer for the DLF Aquifer American Social History Online project. The remainder of his time is spent providing technical support for various ongoing projects at UIUC, including being the developer of the UIUC OAI Registry, providing technical leadership for the library's NDIIPP ECHO DEPository grant project, and various internal projects. Tom's start in digital libraries was as lead developer on the library's NSF-funded Digital Library Initiative (DLI I) project and on the CNRI-funded DLib Test Suite projects.

Janet Eke served as project coordinator of the NDIIPP-funded digital preservation projects based at the University of Illinois at Urbana-Champaign from Fall 2004 through August 2008. She is now the research services coordinator at the Graduate School of Library and Information Science (GSLIS) at the University of Illinois at Urbana-Champaign where she helps to develop services, tools, and resources to support and promote the research efforts of the school. Previously at GSLIS she provided research services at a fee-based custom research unit and taught a master's course in online searching. Before joining GSLIS in 1998, she worked for many years in public libraries.

Matt Cordial is digital library software engineer for the Informatics and Cyberinfrastructure Services unit at the Arizona State University. He is responsible for the design, implementation, and maintenance of applications that interface with ASU's Fedora repository and other external systems. He develops workflow tools for the capture, ingest, and update of a wide range of digital objects: textual objects, images, audio and video data, numeric and geospatial data, etc. Cordial holds a master's degree from the Graduate School of Library and Information Science at the University of Illinois at Urbana-Champaign.

Bill Ingram is a visiting research programmer at the Grainger Engineering Library Information Center at the University of Illinois at Urbana-Champaign, where he provides programming support for several of the library's grant-funded research projects. Recent projects include the NDIIPP-funded Hub and Spoke Tool Suite, the IMLS Digital Content Gateway, and a Mellon-funded experiment using OAI-ORE Resource Maps to support scholarly annotation of digitized books. In addition, Ingram attends the Graduate School of Library and Information Science at the University of Illinois at Urbana-Champaign, where he expects to receive his MS in December 2008.

Robert Manaster is a research programmer for the IT Infrastructure and Software Development Group at the University of Illinois Champaign-Urbana. He is involved in various software development projects, such as BeanCounter (a Web-based library reporting system) and digiTrak (a Web-based mass digitization tracking system. He customizes and integrates vendor software, as well as does database design and consulting. In addition, he provides support and administration for existing IT infrastructure. Robert holds an MS in Library and Information Science from the University of Illinois.