Int J Digit Libr (2007) 7:35–52
DOI 10.1007/s00799-007-0016-7

REGULAR PAPER

Pathways: augmenting interoperability across scholarly repositories

Simeon Warner · Jeroen Bekaert · Carl Lagoze · Xiaoming Liu · Sandy Payette · Herbert Van de Sompel

Published online: 15 May 2007

Abstract In the emerging eScience environment, repositories of papers, datasets, software, etc., should be the foundation of a global and natively-digital scholarly communications system. The current infrastructure falls far short of this goal. Cross-repository interoperability must be augmented to support the many workflows and value-chains involved in scholarly communication. This will not be achieved through the promotion of single repository architecture or content representation, but instead requires an interoperability framework to connect the many heterogeneous systems that will exist.

We present a simple data model and service architecture that augments repository interoperability to enable scholarly value-chains to be implemented. We describe an experiment that demonstrates how the proposed infrastructure can be deployed to implement the workflow involved in the creation of an overlay journal over several different repository systems (Fedora, aDORe, DSpace and arXiv).

1 Introduction

The manner in which scholarly research is conducted is changing rapidly. This is most evident in Science and Engineering [42], but similar revolutionary trends are becoming apparent across disciplines [43]. Improvements in computing and network technologies, digital data capture techniques, and powerful data mining techniques enable research practices that are highly collaborative, network-based, and data-intensive. Moreover, the notion of a unit of scholarly communication is changing fundamentally. Whereas in the paper world, the concept of a journal publication dominated the definition of a unit of communication, in the emerging eScience environment, units of communication are increasingly complex digital objects. The digital objects can aggregate datastreams with both a variety of media types and a variety of intellectual content types, including papers, datasets, simulations, software, dynamic knowledge representations, machine readable chemical structures, etc. Repositories that host such complex digital objects are appearing on the network at a rapid pace.

In the light of these profound changes, we envision the emergence of a natively digital scholarly communication infrastructure that has this wide variety of repositories as its foundation. This infrastructure would leverage the value of the digital objects in the underlying repositories by making them accessible for use and re-use in many contexts. In this infrastructure, repositories are not regarded as static nodes in a scholarly communication system that are merely tasked with archiving digital objects that were deposited there by scholars. Rather, repositories are perceived as the building blocks of a global scholarly communication federation in which each individual digital object can be the starting point of value chains with global reach.

Implementation of this infrastructure brings up a variety of intriguing prospects and associated questions across the whole sociological-economical-legal-technical spectrum. In the Pathways project, a joint project between Cornell University and the Los Alamos National Laboratory, we are

S. Warner (✉) · C. Lagoze · S. Payette
Computing and Information Science, Cornell University,
Ithaca, NY 14853, USA
e-mail: simeon@cs.cornell.edu

J. Bekaert
Faculty of Engineering, Ghent University,
Jozef Plateaustraat 22, 9000 Gent, Belgium

X. Liu · H. Van de Sompel
Research Library, Los Alamos National Laboratory,
Los Alamos, NM 87545, USA

exploring the technical problem domain. We focus on identifying and specifying the fundamental components required to facilitate the emergence of a natively digital, repository-based scholarly communication system. Our research tries to find the appropriate level of cross-repository interoperability that will provide a sufficiently functional technical basis for the realization of the vision, and will stand a realistic chance of being implemented in existing and future repository systems.

This work is important because the current level of cross-repository interoperability is inadequate to support advanced forms of communication. Different communities have followed their own perspectives on repository design, implementation and management as well as on digital object representation and identification. Current interoperability is provided mainly by support of the OAI-PMH [34] and its mandatory Dublin Core metadata format [20]. Realizing the vision will require significantly augmented cross-repository interoperability.

The remainder of this paper is organized as follows. Section 2 presents several motivating scenarios, and then Sect. 3 describes related interoperability work. The following sections then introduce ideas for a cross-repository interoperability framework that have resulted from the Pathways project. The proposed high-level requirements for participating repositories can be summarized as follows:

- Support for a shared *data model* for *digital objects* (Sect. 4.1).
- Support for a *surrogate* format that serializes the *digital object* in accordance with the *data model* (Sect. 4.2).
- Support for three core repository interfaces: *obtain*, *harvest* and *put*, to allow dissemination and ingest of *surrogates* (Sect. 5).

The proposed framework also requires a shared service registry that lists the network location of the core interfaces for participating repositories. Section 6 describes the service registry and possible format and semantic registries that would further empower the environment. In Sect. 7 we describe experiments to implement an overlay journal scenario (an example we will use repeatedly throughout this paper) using this framework over existing repositories. Section 8 presents plans for future work, and Sect. 9 draws some conclusions. A less technical exposition of these ideas is given in [16].

2 Motivating context

In order to gain insights into the characteristics of the desired interoperability framework, it is helpful to investigate scenarios that drive this need for augmented interoperability. We see two classes of cross-repository value-chains: rich cross-repository services and cross-repository scholarly communication workflows.

In the first class of cross-repository value chains, repositories are regarded as sources of materials that can be used in services with a reach beyond the boundaries of a single repository. Materials should be exposed by repositories in a manner that allows for the seamless emergence of rich and meaningful services. Discovery services are an obvious example of this class, and, although support of the OAI-PMH has resulted in a suite of cross-repository discovery capabilities, their functionality remains limited. For example, imagine creating a special-purpose search engine that collects only machine-readable chemicals structures, expressed using the XML Chemical Markup Language (CML), contained in digital objects hosted by repositories worldwide. The current interoperability environment provides neither the ability to expose digital objects at a repository interface in a manner that unambiguously reveals the digital object's constituent datastreams, nor the language to express their intellectual content type (e.g. chemical structure). As a result, the creation of the cross-repository chemical search engine would currently be truly complex, and would involve numerous repository-specific trial and error procedures.

Consider the case where monitoring agencies make semantically tagged data on Arctic sea ice available in interoperable repositories. An automated alerting service might then be able to discover and use both raw and processed data (with raw data provenance accurately indicated) to provide early warning of events such as the abrupt shrinkage in Arctic sea ice in 2005. The output might be a report, a new digital object, containing both static 'snapshot' results and importing dynamically computed elements. Accurate versioning of datasets would allow readers to be made aware of later amended inputs and perhaps even to recompute the results included in the report based on machine-actionable descriptions of the transform and visualization service. A newspaper article on the findings might reference the source reports allowing readers to delve into and understand the sources and the basis of the claims as far as their understanding permits.

In the second class of cross-repository value chains, repositories are regarded as the basic building blocks of a digital communication system, and scholarly communication itself is seen as a global cross-repository workflow [18]. Digital objects contained in repositories are the subjects of the workflows, and are used and re-used in many contexts.

Citation is probably the most obvious example of this. In today's scholarly communication system, citation is implemented by inserting textual information describing a cited paper at the end of the citing paper, either by just typing it, by copy/pasting it from a Web page, or by importing metadata from a personal bibliographic citation tool. Thus, citations that are included in a digital manuscript are purely textual and are not natively machine readable or machine actionable. As a result, various *post-factum* approaches have been devised to connect citing paper to cited paper by means of hyperlinks in the Web environment [13]. These approaches include fuzzy metadata-based citation matching [23], the DOI-based CrossRef linking environment [11], and the OpenURL framework for context-sensitive linking [14]. The variable quality of citation metadata, among other factors, means that none of these approaches is foolproof. Furthermore, it is challenging to imagine how these approaches would extend beyond conventional scholarly papers, into the realm of complex digital objects that contain datasets, simulations, visualizations and so forth. It is therefore intriguing to think about citation as the re-use of the cited digital object in the context of the citing digital object.

To understand this expanded view of citation, imagine being able to drag a machine readable representation of a digital object hosted by some repository, and to drop it into the citing object that, once finalized, is submitted into another repository. Now imagine being able to do the same for the citing object *ad infinitum*. Assuming that the machine readable representations that are being dragged and dropped contain the appropriate properties, the result would be a natively machine-traversable citation graph that would span across repositories worldwide. With appropriate user tools this would not only be vastly more functional than current forms of citation, but also simpler to use and to manage.

Collectively, these scenarios lead to a number of high-level observations:

Long-term perspective — Scholarly communication is a long-lasting endeavor, and, as a consequence, a long-term perspective should inspire the thinking about a future digital scholarly communication infrastructure. Clearly, this yields requirements related not just to the longevity of repositories and their collections, but also to the interoperability framework. The framework should be defined with sufficient abstraction to allow implementation using different technologies as time goes by, and should not be tied to a specific type of identifier, but rather support all current and future identification systems.

Content-transfer is often unnecessary — Most of the value chains illustrated in the above scenarios do not require the transfer of all digital object content. Instead just a subset appropriate to the particular value chain. For example, the citation scenario requires only the transfer of the bibliographic metadata of the cited paper, whereas the search engine scenario only requires the transfer of the chemical formula. Full content-transfer as required for repository mirroring is just one of many use cases that should be enabled by a desired solution.

Fine grained identification — Identifiers of journal articles, such as DOIs, are typically repository independent in the sense that copies of a paper with a given identifier stored by multiple repositories share the same public identifier. This level of identification granularity is sufficient for citation purposes. However, it becomes inadequate when trying to record the chain of evidence for cross-repository value chains because these have a specific digital object from a specific repository as their subject. This means that a finer level of identification granularity is required than provided by the existing bibliographic infrastructure.

3 Related work

Pathways is focused on defining a common *data model* and *service interfaces*. These are designed to enable re-use and re-combination of digital objects and their components, to facilitate workflows over distributed repositories, and to enable computation and transformation of digital objects with dynamic service linkages. A key aspect of this work is that it explicitly handles the notion of provenance or lineage when content is re-used.

A significant amount of work exists in the design and specification of data models for digital objects, and in the creation of XML representation formats to promote the interoperable transmission and exchange of digital objects. XML representation formats include the Metadata Encoding and Transmission Standard (METS) [37], the MPEG-21 Digital Item Declaration Language [8], the IMS Content Packaging XML Binding [26], and the XML Formatted Data Unit (XFDU) [45]. Many of these formats have been used to enable the transfer of digital assets among systems. A notable example is the use of MPEG-21 DIDL in the transfer of the American Physical Society collections to Los Alamos National Laboratory [6].

There is no doubt that multiple data models for complex objects exist and will continue to be favored by different communities. The challenge is to develop a simple and flexible overlay data model that does not depend upon asset transfer, and can accommodate the essence of these different content models, yet can provide a simple low-barrier entry point for interoperability among repositories.

The Content Object Repository Discovery and Registration/Resolution Architecture (CORDRA) [33] is similar to Pathways in its goal to provide an open, standards-based model for repository interoperability. However, CORDRA is primarily focused on enabling interoperability between learning object repositories via federated registries of metadata catalogs. Unlike Pathways, CORDRA is specified upon a landscape of authored metadata. The Pathways data model is built upon a generic, graph-based abstraction that does not prescribe specific metadata other than small set of key attributes for objects. While CORDRA offers support for retrieval of content, Pathways addresses both retrieval and write for complex objects among heterogeneous repositories. Finally, while a distributed name resolution system (e.g., the handle system) is a necessary architectural pillar in CORDRA, the Pathways identifier scheme does not depend on a shared digital object identifier resolution service shared by distributed repositories.

There are a number of other projects in the higher education community devoted to the goal of repository interoperability within service-based architectures. Similarly motivated work is being done with the EduSource Community Layer (ECL) [21], the DLF Asset Action Experiments [19], the Open Knowledge Initiative Open Service Interface Definitions (OSIDs) [40]. These projects each specify a middleware service layer to enable applications to be built over heterogeneous data sources. Pathways is distinguished in that it is focused on defining a minimal set of read/write services necessary to enable access and re-use of complex objects in a distributed, heterogeneous repository environment. The intent of Pathways is to specify relatively lightweight services that are easy to deploy over existing architectures. At the same time, Pathways is motivated to provide a model that can record and exploit provenance relationships as content is re-used across different services.

The challenge of service-based repository interoperability is being taken up by many other communities, often with different definitions of the basic concept of a "repository". In terms of service interfaces and APIs, repository interoperability is being addressed both from an access perspective and an authoring perspective (i.e., write, put). Many efforts are positioned around a limited view of "content", typically individual content byte streams (an image, a web page, a PDF document), or hierarchies of content byte streams with simple descriptors.

For example, there are many new services that position the web as both a readable and writable space, albeit in a limited manner. Atom [3] provides an API for an application level protocol for publishing and editing web resources. It also provides an XML data format that can be used in both the syndication and authoring of content. The Amazon S3 [2] web service provides an interface to support reading and writing, ultimately providing an internet data storage service that is scalable, reliable, and fast. SRW Search/Retrieve and Update [41] defines a web-service interface for retrieving and updating metadata records. Web-based Distributed Authoring and Versioning (WebDAV) [44] enables the web servers to be exposed as writable, in addition to readable, by providing an interface for uploading content using a file and directory paradigm. Each of these services share with Pathways the notion of simple web-based interfaces for creating and accessing content over the web. However, a key distinction of Pathways is its focus on complex digital objects as units of content as compared to single-content byte streams (e.g., a file). Another distinction of the Pathways work is that it is primarily intended to be an interoperability model for managed repositories, as distinguished from more nebulous storage services on the open web.

Pathways employs a graph or tree-based data model to overlay heterogeneous data sources, which is also the basis of several other efforts. Recently, JSR 170 [30] has garnered much attention. This is a specification of a Java-based API for interacting with heterogeneous "content repositories" and repository-like applications in a uniform manner. The basic metaphor for interaction is that of a hierarchy of nodes with properties, where node properties can be either simple data types or binary streams. JSR 170 is positioned similarly to how JDBC is for relational databases. It is most useful for developing Java-based applications with a standard interface for connecting with content storage components (i.e., "content repositories"). Since it is not web services oriented, it is not clear the impact it could have in providing interoperability among distributed institutional repositories, and in non-Java environments.

The Pathways framework is intended to be consistent with existing and emerging web architecture principles and should be easily implemented using existing web protocols and standards. In considering the W3C recommendation for the Architecture of the World Wide Web [28], the Pathways framework has been influenced by the need for URI-based identifiers for resources, the notion that resources can have one or more "representations", and that these representations can be sent or received via simple protocols. Pathways is influenced by work in the semantic web community, particularly the Resource Description Framework (RDF) [32] as a model for expressing resources in a graph-oriented manner as resource nodes with property and relationship arcs.

4 Digital objects, the Pathways Core data model, and surrogates

The goal of our work on data models and interfaces is the creation of an interoperability layer, as indicated in Fig. 1. We expect that this layer will overlay data models and service interfaces that are distinct to individual repository implementations. These repository-specific models and interfaces may provide functionality outside and above the models and interfaces described here, which are intended to represent the intersection (rather than union) of individual repository features.

We use the following definitions:

Digital object — In the manner of the seminal Kahn and Wilensky paper [31] we use the notion of a digital object to describe compositions of digital information. This is purposely abstract, and is not tied to any implementation

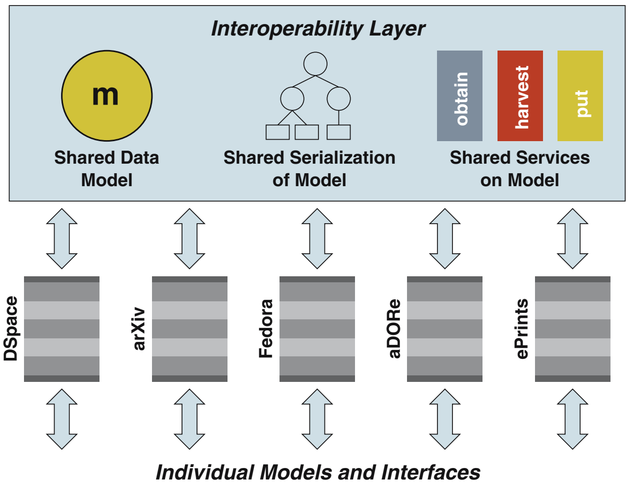

Fig. 1 Interoperability layer over heterogeneous repositories

or data model. The principal aspects of a digital object are digital data and key-metadata. Digital data can be any combination and quantity of individual datastreams, or physical streams of bits, and can consist of nested digital objects. Key-metadata, at a minimum, includes an identifier that is a key for service requests on the digital object at a service point.

Data model — We describe a data model, the Pathways Core, that provides a formalization for overlaying digital objects on a network of heterogeneous repositories and services. We use UML to describe this data model, but it could be described in other formalizations such as XML schema or OWL.

Surrogate — We use the term surrogate to indicate concrete serializations of digital objects according to our data model. The purpose of this serialization is to allow exchange of information about digital objects from one service to another and thus propagate them through value chains. We use RDF/XML for constructing our surrogates, because it is useful for representing arbitrary sub-graphs.

The primary goal of the data model, and consequently the surrogates that represent it on the wire, is not asset or content-transfer. Rather we have designed the data model primarily as a framework that describes the abstract structure of information objects, and the properties of that abstract structure such as lineage, identity, and semantics. The linkage in the model from the abstract structure to the physical content is *by-reference* rather than *by-value* containment.

There are a number of good reasons to not mandate asset transfer in the interoperability fabric. Full asset transfer is necessary for only a subset of possible applications. One notable one is preservation mirroring, and thus preservation frameworks such as the Reference Model for an Open Archival Information System (OAIS) [39] include the notion of information packages that imply full transfer of information units. Many other applications such as the overlay journal example described later can be accommodated without the overhead of shipping all the bits between repositories and services. By supporting by-reference content, the Pathways Core model enables services to selectively decide when and if to dereference and pull content into the service environment. This promotes the notion of "service-tuned" asset transfer, where each service can be configured to respond to by-reference content in a manner appropriate to the context.

In a number of cases, full asset transfer is forbidden or undesirable. For example, a rights holder may be willing to allow inclusion of their asset in another context by means of reference through a surrogate, but may be unwilling to transfer the datastream itself. Or, the rights holder might allow the asset transfer if the assets are placed in some digital rights management (DRM) wrapper.

Finally, static transfer of an asset may be undesirable in the case of dynamic information objects, such as data sets derived from sensor networks. We foresee a number of applications in the scholarly domain where such dynamic objects are desirable, such as astronomy publications that include the latest sky survey data.

4.1 Pathways Core data model

The Pathways Core data model is based on the notion of a graph of abstract *entities* with concrete *datastreams* as leaves. In this model, a digital object is a sub-graph rooted at an entity. The data model is designed to meet the following requirements:

1. It permits recursion for arbitrary levels of *entity* containment.
2. It provides an explicit link to the concrete representation, or component *datastreams*, of the digital object.
3. It includes a notion of object identity that is independent of specific identifier schemes.
4. It expresses lineage among objects, providing evidence of derivation and workflow among objects.
5. It accommodates the linkage of semantic tags to information entities that extend the functionality of format tags to the domain of complex, multi-part objects.
6. It allows the maintainer of the object to assert persistence of the availability of a surrogate.

A UML structure diagram of the Pathways Core is shown in Fig. 2. The correspondence of features of the model to the requirements list above is indicated by the numbered properties. Each feature of the model is explained in more detail in the following sections. Our goal has been to find the minimal set of features necessary, the core properties. Certain uses or applications may require refinement of these rela-

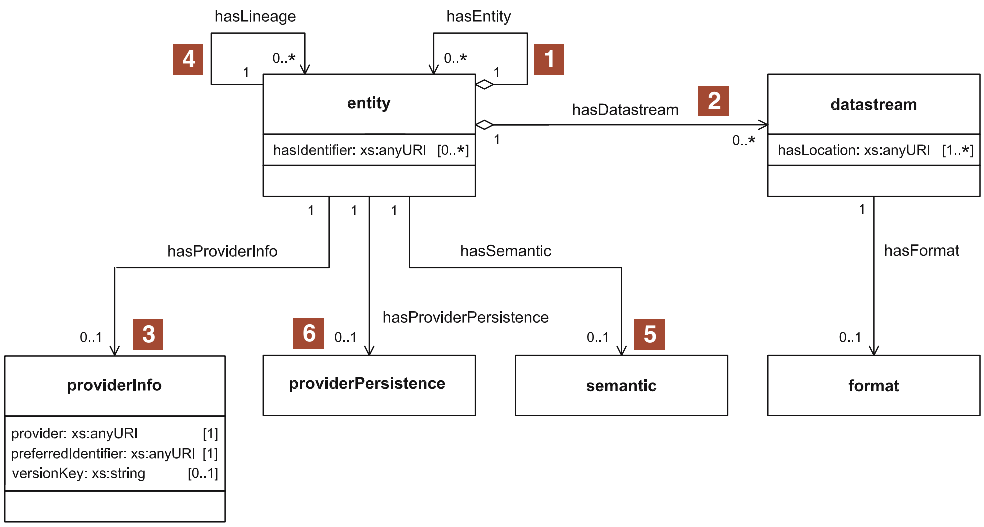

Fig. 2 UML diagram of the Pathways Core data model with parts that fulfill particular requirements numbered

tionships or the addition of new relationships, and we believe that such extensions can be added without breaking the core functionality.

4.1.1 Entity recursion

At the root of the Pathways Core is the notion of an *entity*. As shown in Fig. 2, this is the attachment point of a set of properties that associate the entity with its required and optional features. One property is hasEntity, which expresses recursive containment of entities. This maps to the Kahn/Wilensky [31] notion that digital objects can contain nested digital objects. An example of the utility of this recursive relationship is modeling of an overlay journal. In this case, a top level entity could represent the journal itself, with semantic, persistence, and identity attributes that correspond to the journal. A journal "contains" issues, which themselves may be entities, with associated properties. This recursion naturally continues, with issues "containing" articles.

As indicated in Fig. 3, an entity is an abstract concept, distinct from concrete datastreams described in the next section. This abstract/concrete distinction is fundamental to the model — removing assertions of identity, persistence, lineage, and semantics from individual physical manifestations

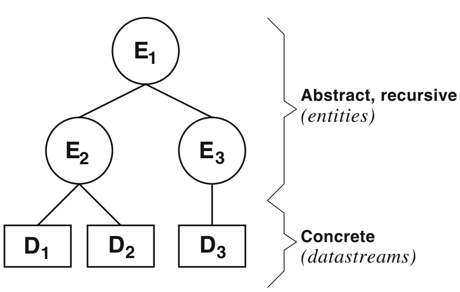

Fig. 3 Entity recursion and concrete representation

of intellectual objects. This separation of abstract and concrete properties (or attributes) is similar to that in the FRBR model [38].

4.1.2 Concrete representation

As indicated in Fig. 2, an entity can have several hasDatastream properties. The motivation for this is well-established in compound document formats such as METS, MPEG-21 DIDL, and Fedora FOXML, which allow a single object to have multiple datastreams with different media types (e.g., the availability of a scholarly paper in PDF, Word and TeX).

A datastream has both a format (e.g. a format registered in GDFR [1] or PRONOM [12]) and a location, a URL to request a dissemination of the datastream. The datastream association is intentionally *by-reference* rather than *by-value*, to avoid mandating asset transfer for the reasons given earlier.

A typical digital object will contain one or more datastreams. The digital object represented in Fig. 3 comprises a top-level entity E_1, with sub-entities E_2 and E_3. The entity E_2 has two datastreams, D_1 and D_2, which might be alternate expressions of the entity E_2. As semantic assertions appear only at the level of the entity, both D_1 and D_2 are assumed to have any semantics expressed for E_2. The entity E_3 has just a single datastreams D_3 and may thus be used to express semantics that apply just to D_3, as separate from the semantic of E_1. For example, E_1 might be a "journal issue" with two articles E_2 and E_3, and the article E_2 happens to be available in both PDF and Word formats.

4.1.3 Identity

We recognize the reality that one identifier technology will never dominate and have thus incorporated two notions of identity. First, the hasIdentifier property allows expression of URIs associated with a digital object, a DOI for example. Second, the hasProviderInfo property introduces a relatively simple repository-centric identifier paradigm which permits precise identification of digital objects in the particular repository, facilitating re-use and accurate provenance records. This paradigm is not intended to replace existing identifier mechanisms or to interfere with future technologies in this area. Rather it is intended as a future-proof long-term scheme that can co-exist with other identifier mechanisms.

The hasProviderInfo property has three components:

- provider — The identity of the repository (i.e.; the service point providing access and ancillary services on the digital object). We assume that the participants in this infrastructure — institutional repositories and the like — have a commitment to of their repository identity. Indirection via a repository identifier presumes some technology for registering repositories and resolution to the location of their service interfaces. Registries are discussed later, in Sect. 6.
- preferredIdentifier — The identity of the entity within the repository. This serves as the key for making service requests upon the digital object at the service point defined by the repository (provider) identity. As explained later in this paper, the basic repository service is a request for a surrogate of the digital object. We expect, however, that a host of other services will evolve. We emphasize that the syntax, semantics, and resolution of the identity of the object is local to the individual repository, rather than being global as in more ambitious identifier schemes.
- versionKey — This is a means of parameterizing a service request on an object according to version semantics. The intention here is to provide an opaque hook into individual repository versioning implementations, rather than assuming or imposing some universal cross-repository version schema.

Two copies of the same object in two different repositories may have the same identifier expressed via the hasIdentifier property. However, they will have different providerInfo because they are available from different repositories.

4.1.4 Lineage

Isaac Newton wrote “If I have seen further it is by standing on the shoulders of Giants” [27]. In the face of massive changes in scholarship since Newton’s time, one constant is the evolution of scholarship, whereby new results are built on the innovations of earlier scholars. We believe therefore that the interoperability infrastructure must support the notion of lineage, natively linking entities to other entities from which they are derived.

As shown in Fig. 2, entities in the model can link to other entities through the hasLineage relationship. This linkage leverages the hasProviderInfo identity of the entity (or entities) from which the new entity derives, thus allowing an entity to express its derivation from another entity and specifically state both the repository origin of the source object and its version semantics. Furthermore, since the model is recursive, entities can contain entities and the derivation of contained parts of objects can be similarly expressed.

This lineage capability is illustrated in Fig.4. The entity labeled E_1 is derived from that labeled E_2. For example, E_1 may be translation of E_2 into a new language. E_2, as illustrated, contains sub-entities with respective derivations from E_3 and E_5. For example, E_2 may be an issue of an overlay journal with articles that are edited versions of the preprints E_3 and E_5, where E_5 is itself a sub-entity of the preprint series E_4. These cases illustrate re-use at different granularities.

The result of these lineage links among entities at the interoperability layer is a web of *evidential citation*. This graph indicates both the workflow origins of an information object — the partial ordering of information objects from which it derives — and also the curatorial heritage of the object — the repositories and services responsible for its legacy. This new, uniquely networked and digital form of citation

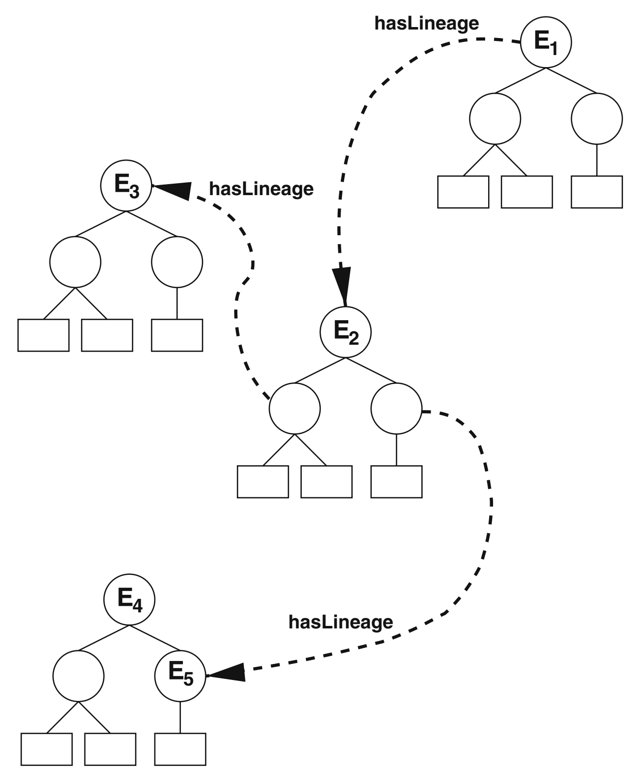

Fig. 4 Relating entities by lineage

provides a finer level of identification than convention bibliographic citation. In the case that a repository has an object derived from an object in another repository, there is a local choice as to whether the same object identifier is used or a new one generated. This choice would be presumably be influenced by repository policy, community agreements and by the kind of value chain implemented. In either case, two observations can be made. First, the providerInfo includes the provider which make the complete identification unique and distinguished the objects. Second, the hasLineage property of the derived entity provides and unambiguous link back to the original entity.

Both cases are illustrated in Fig. 5, which shows the entity labeled E_1 taking part in value chains that result in new entities, E_2 and E_3, in different repositories. In all cases the entities are uniquely identified by the providerInfo, even though E_2 has the same preferredIdentifier as E_1. Also, both E_2 and E_3 indicate their lineage from E_1 with the providerInfo extracted from E_1.

We imagine that the hasLineage relationship is a superclass of the many types of inter-entity derivation relationships that could be expressed. Thus, future evolution of the infrastructure might refine this relationship.

4.1.5 Semantics

We envision applications that need to know about the “semantic” composition of digital objects in addition to knowing the media-format types of the individual datastreams. A complex digital object might represent a “dissertation” or a “journal article”, each of which might have datastreams that are images, data sets, spreadsheets, or text in various formats.

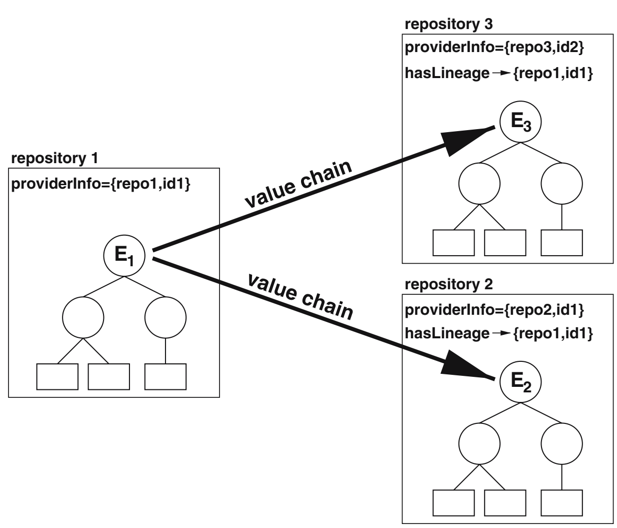

Fig. 5 Identification and lineage of derived entities in different repositories. The shorthand {provider,preferredIdentifier} is used for providerInfo, and versionKey is omitted

One particularly interesting application is service matching. The utility of automated match of preservation services to information objects has been demonstrated by the PANIC work [24]. While PANIC demonstrates the utility of automation for individual datastreams based on media type, we would like to enable similar services over complex objects and based on intellectual content types.

The Pathways Core therefore associates the hasSemantic property with each entity. The target of this property is a URI specifying the semantic typing of the entity. Admittedly, no universal semantic registry exists at this time. However, the property could be exploited by individual communities that develop local schemes, and later extended to more widespread use.

4.1.6 Persistence

The history of persistence of information artifacts, especially digital objects, is riddled with examples of the gaps between intention, expectation, and reality. Despite our best intentions to provide storage of and access to digital information “forever” (or even a few months!), the realities of hardware failures, format rot, and mismanagement frequently interfere. This must be considered in the design of any information interoperability framework.

Therefore, we have taken a purposely modest approach to persistence that is oriented towards surrogates and services over surrogates, rather than towards digital objects. The hasProviderPersistence property associated with an entity is a slot in which the repository can declare, by means of a URI, the longevity of its commitment towards providing services over the respective entity. The repository making this commitment is identified as the provider in the entity’s hasProviderInfo property. Since the core service in the interoperability fabric is the dissemination of a surrogate for the entity, hasProviderPersistence indicates the level of commitment of the respective repository to provide access to a surrogate for the entity. While there is clearly scope for subtle refinement of persistence declaration, at this point we propose a set of just two persistence declarations:

- The entity is transient and the repository makes no commitment to providing services for it over time.
- The entity is persistent and the repository intends to respond to service requests for it over time.

4.2 Surrogates and serialization

An individual instance of the Pathways Core data model, a representation of an individual digital object, is packaged and transmitted as a surrogate: a serialization that conforms to the data model. We note possible terminological confusion here but have not found a word with less baggage. By surrogate we mean a serialization that substitutes for the digital object and must therefore reveal all essential characteristics, and is thus distinguished from some arbitrary representation. The obtain and harvest interfaces (described in sects. 5.1 and 5.2) provide the means for clients to request a surrogate. Similarly, a put request (described in Sect. 5.3), which requests deposit of a digital object in a repository, contains a surrogate as a payload.

We have found that RDF [32] is a useful tool for modeling the graph-like structure of information in the Pathways Core. We have done this by associating URIs with the properties in the Pathways Core and similarly associating URIs with a number of controlled vocabularies such as persistence, formats, and semantics that are the values of Pathways Core properties. RDF modeling naturally led to the adoption of the RDF/XML syntax [4] as the serialization syntax for Pathways Core surrogates. A fragment of an example of this syntax is shown in Fig. 6.

5 Repository interfaces: obtain, harvest and put

We have described the Pathways Core data model and a surrogate that serializes the model. For these to enable repository interoperability, a set of essential services are required. Three repository interfaces with the following functions fulfill this need and are described below:

- An *obtain interface* which, in its most basic implementation, allows the request of a surrogate for an identified digital object from a repository.
- A *harvest interface* that exposes surrogates for incremental collection or harvesting.
- A *put interface* that supports submission of one or more surrogates into the repository, thereby facilitating the addition of digital objects to the collection of the repository.

5.1 Obtain interface

Pathways defines an obtain interface that supports the request of services pertaining to an identified digital object within a repository. The simplest implementation of the obtain interface allows requesting a surrogate for an identified digital object from a repository. Such an interface can be regarded as an identifier-to-surrogate resolution mechanism that resolves the preferredIdentifier of a digital object into a surrogate of that digital object.

The information needed to construct an obtain request is recorded in the providerInfo property of the surrogate itself. The providerInfo is a triple consisting of the identifier of the repository that exposes the surrogate, the preferredIdentifier of the digital object and an optional versionKey. By using the identifier of the repository, the location of the obtain interface of the identified repository can be found by a look-up in a service registry (see Sect. 6). Once known, one can use the preferredIdentifier of the digital object (and the optional versionKey) to obtain a surrogate using the repository's obtain interface.

Higher levels of the obtain functionality have been explored theoretically by Bekaert [5]. Straightforward

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:core="info:pathways/core#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <core:entity rdf:about="info:pathways/entity/info%3Asid%2Flibrary.lanl.gov%3Apathways/info%3Adoi%2F10.1016%2Fj.dyepig.2004.12.010">
    <core:hasSemantic rdf:resource="info:pathways/semantic/journal-article"/>
    <core:hasIdentifier>info:doi/10.1016/j.dyepig.2004.12.010</core:hasIdentifier>
    <core:hasProviderPersistence rdf:resource="info:pathways/persistence/persistent"/>
    <core:hasProviderInfo>
      <core:providerInfo>
        <core:preferredIdentifier>info:doi/10.1016/j.dyepig.2004.12.010</core:preferredIdentifier>
        <core:provider>info:sid/library.lanl.gov:pathways</core:provider>
      </core:providerInfo>
    </core:hasProviderInfo>
    <core:hasEntity>
      <core:entityrdf:about="info:pathways/entity/info...(shortened)...lanl-repo%2Fssm%2Fdoi-10.1016%2Fj.dyepig.2004.12.010">
        <core:hasSemantic rdf:resource="info:pathways/semantic/bibliographic-citation"/>
        <core:hasIdentifier>info:lanl-repo/ssm/doi-10.1016/j.dyepig.2004.12.010</core:hasIdentifier>
        <core:hasProviderPersistence rdf:resource="info:pathways/persistence/persistent"/>
        <core:hasProviderInfo>
          <core:providerInfo>
            <core:preferredIdentifier>info:lanl-repo/ssm/doi-10.1016/j.dyepig.2004.12.010</core:preferredIdentifier>
            <core:provider>info:sid/library.lanl.gov:pathways</core:provider>
          </core:providerInfo>
        </core:hasProviderInfo>
        <core:hasDatastream>
          <core:datastream>
            <core:hasFormat rdf:resource="info:pathways/fmt/pronom/1000"/>
            <core:hasLocation>http://purl.lanl.gov/demo/adore-arcfile/00e682eb-a87eb27b0c79</core:hasLocation>
          </core:datastream>
        </core:hasDatastream>
        ...
```

Fig. 6 Excerpt from a sample surrogate that serializes the Pathways Core in RDF/XML extension of the obtain concept allows the request of any supported service pertaining to an identified digital object. This includes the request of services pertaining to datastreams of the digital object. Possible examples are requests to obtain a surrogate of an identified article, requests to obtain a PDF datastream of that same article, requests to obtain an audio version of that article by applying a text-to-speech service upon the PDF datastream, and so forth. Such services can be considered a superclass of the basic obtain functionality described above, and do not have to be supported by all repositories. Rather, such services would typically be supported by autonomous service applications that overlay one or more repositories and use surrogates that are obtained through interaction with the core obtain interface of the underlying repositories.

One technology that lends itself for implementing these obtain interfaces is the OpenURL Framework for Context-Sensitive Services [46]. The OpenURL standard originates from the scholarly information community where it was proposed a solution to the provision of context-sensitive reference links for scholarly works such as journal articles and books [14]. The initial standard was generalized to create the current NISO OpenURL Framework which describes a networked service environment, in which packages of context information (ContextObjects) are used to request context-sensitive services pertaining to a referenced resource. Each ContextObject contains various types of information that are needed to provide context-sensitive services. Such information may include the identifier of the referenced resource, the Referent, the type of service that needs to be applied upon the Referent (the ServiceType), the network context in which the resource is referenced, and the context in which the service request takes place.

In this way, the core obtain interface can be implemented as an OpenURL Framework Application. The ContextObject used in the obtain request conveys the following information:

A Referent — The digital object for which an obtain request is formulated. The Referent is described by means of its preferredIdentifier.

A ServiceType — The service that generates a surrogate of the identified digital object.

Beyond meeting our basic requirements, the OpenURL Framework has the following attractive properties:

- it makes a clear distinction between the abstract definition of concepts and their concrete representation and the protocol by which such representations are transported. A ContextObject may be represented in many different formats and transported using many different transport protocols, as technologies evolve. Yet, the concepts underlying the OpenURL Framework persist over time.
- it does not make any presumptions about the identifier namespace used for the identification of digital objects (or constituents thereof), and hence, provides for an obtain interface that can be implemented across a broad variety of repository systems.
- it allows information about the context in which the obtain request took place to be conveyed. This information may allow delivery of context-sensitive service requests. Of particular interest is information about the agent requesting the obtain service (the Requester). This information could convey identity, and this would allow responding differently to the same service request depending on whether the requesting agent is a human or machine. Similarly, different humans could receive different disseminations based on recorded preferences or access rights. The OpenURL Framework is purposely generic and extensible, and would also support to convey the characteristics of a user's terminal, the user's network context, and/or the user's location via the Requester entity. Though, this type of context-related tuning may not be important when requesting surrogates of digital objects, it may prove to be essential when requesting rich services pertaining to datastreams.

5.2 Harvest interface

A harvest interface allows collecting or harvesting of surrogates of digital objects. In addition to the facility to harvest all the surrogates exposed by a repository, we believe it is necessary to provide a facility allowing some forms of selective harvesting. The simplest, and perhaps most useful, form of selective harvesting is to allow downstream applications to harvest surrogates only for those digital objects that were created or modified after a given date. This echoes the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [34] with the same motivation: downstream applications may need an up-to-date copy of all the surrogates from a repository in order to provide some service, and incrementally harvesting surrogates of newly added or modified digital objects is an efficient way to do this.

A harvest interface could be implemented using various technologies such as the OAI-PMH, RSS or Atom, or with a subset of more complex technologies such as SRU/SRW. The OAI-PMH is a well established harvesting technology within the digital library community and allows aggregation of metadata from compliant repositories using a datestamp-based harvesting strategy. Although the OAI-PMH was first conceived for metadata harvesting, it can be used to transfer any metadata or data format, including complex-object formats, expressed in XML according to an XML Schema [17]. The OAI-PMH is thus capable of providing the harvest functionality, and the ability to leverage existing OAI-PMH implementations is a significant benefit.

To support the harvest interface, the underlying OAI-PMH interface must follow these conventions:

- Each OAI-PMH item identifier must match the preferredIdentifier of the Pathways Core digital object. This avoids the need for clients to record relationships between OAI-PMH identifiers and digital object identifiers which can become complex in various aggregation scenarios.
- The OAI-PMH datestamps must be the datetime of creation or modification of the digital objects as discussed in [17].
- It must provide a metadata format for surrogates as described in Sect. 4.2.

It is worth noting one possible issue. The OAI-PMH specification is bound to the HTTP protocol and the XML syntax for transporting and serializing the harvested records. While this approach proves to be satisfactory in the current technological environment, it may prove to be inadequate as technologies evolve. If this work were to be tightly bound with the OAI-PMH then an abstract model would need to be created. However, if OAI-PMH is used simply as one possible technology to implement harvest functionality then it could later be replaced.

5.3 Put interface

Pathways defines a put interface to promote interoperable transmission of surrogates to one or more target digital repositories. As with the obtain interface, digital objects are expressed as surrogates. At the interface level, put operations are simple and unassuming. They can be understood as a *request for deposit* of a digital object. This distinguishes the put interface from similar operations found in other push-oriented services whose purpose is to facilitate upload of binary content streams, or transfer of assets using community-specific content packages such as METS, IMS-CP, or MPEG-21 DIDL.

The put interface does not presuppose that target repositories conform to any particular underlying storage scheme (e.g., hierarchal file system, a web server with directories, relational database, etc.). Additionally, the put interface is neutral about the underlying data model of target repositories. The only requirement is that digital objects be represented as surrogates expressed in the Pathways Core, which is specifically designed to transcend the particulars of heterogeneous data models. The graph-based nature of this model provides the flexibility to support the submission of both simple and complex digital objects.

The put interface, in combination with a surrogate, is intended as a means for transmitting just enough information to enable a receiving repository to make decisions on how to process a surrogate — without anticipating or assuming an underlying repository's requirements for ingest. Datastream content is expressed by-reference in the surrogate, via the location property. With this constraint, a surrogate represents a "shallow copy" of a complex object since there is no transmission of raw content within the surrogate. As discussed earlier (Sect. 4), this constraint is motivated by the need for simplicity, and the desire to keep authentication and authorization concerns out of the functional definition of the put interface. Authentication, authorization and policy are expected to be handled at service implementation layers.

Unlike the obtain and harvest interfaces, no protocol or technology stands out as an obvious implementation option for the put interface.

6 Registries

The proposed interoperability framework requires at least one supporting infrastructure component: a service registry to associate providers with services. Additional format and semantic registries would significantly enrich the environment. No particular technical implementation is implied by the use of the term registry, but rather the general ability to record, share and retrieve terms of a controlled vocabulary alongside their associated properties.

6.1 Service registry

A service registry is fundamental to the framework, as it facilitates locating the core service interfaces of participating repositories. This registry has the identifier of a repository (provider from providerInfo in the Pathways Core) as its primary key, and it minimally stores the actual network location of the obtain, harvest and put services, where supported. Thus, given a surrogate with providerInfo (provider, preferredIdentifier, versionKey), it is possible for an application to use provider as a look-up key in the service registry to retrieve the location of the core service interfaces for the repository identified by provider. Once this information is available, actual service requests can be issued against those interfaces. For example, in order to retrieve an up-to-date surrogate, the application can issue an obtain request using the preferredIdentifier and optional versionKey of the providerInfo as shown in Fig. 7. The use of a registry permits repository interfaces to change their network location, allows different services (including those not yet imagined) to be associated with a repository, and makes the combination of repositories trivial.

It should be noted that, in contrast to other repository federation approaches such as CORDRA [33], ADL-R [29], aDORe [15], and the Chinese Digital Museum Project [10], the proposed framework does not require a registry of all

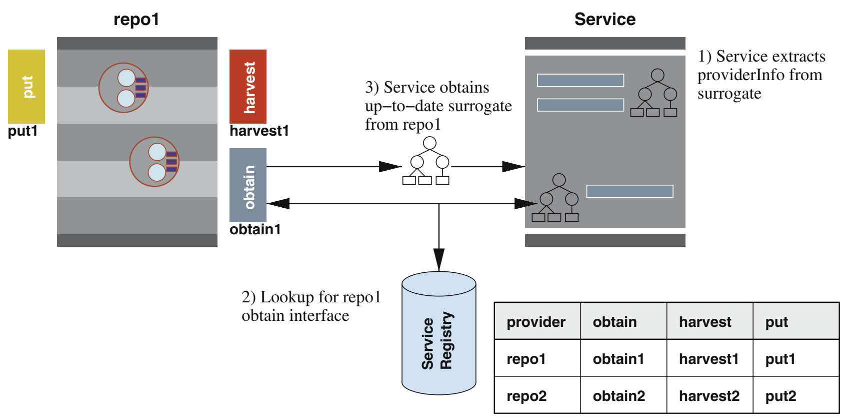

Fig. 7 Use of the service registry

digital objects in all contributing repositories thereby allowing location of a digital object given its identifier. In the proposed framework, a surrogate carries its self-identifying providerInfo, which, through the intermediation of the service registry, allows location of the service interfaces of the originating repository. This approach alleviates two major drawbacks inherent in the use of digital object registries. First, given an identifier, how does one know that it is an identifier of a digital object from repositories contributing to the federation, and hence that a look-up in the federation's object registry is meaningful? It seems that this question can only be answered if all repositories in the federation share a common, recognizable identifier scheme. This is a significant requirement, especially in light of the considerations regarding the long-term horizon of desired solutions. Second, the scale of object registries is several orders of magnitude larger than that of the proposed service registry because the latter only contains an entry per repository, not per digital object. The repercussions for operating the registry infrastructure are obvious.

6.2 Format and semantic registries

While the service registry is essential for the operation of the proposed framework, two other registries, while less fundamental, would significantly enrich the functionality of the anticipated environment.

First, it is now widely recognized that repositories, especially in preservation environments, must support more finely grained identification of digital media formats than is provided by MIME types. A format registry that has the identifier of a digital format as its primary key and that records various properties of the format have been proposed by both the PRONOM [12] and GDFR [1] efforts. Format identifiers would be used for the format property available at the datastream level of surrogates. Such a fine level of format identification would, for example, enable rich format-based service matching as explored in the PANIC [24] and aDORe [7] efforts.

Second, automated object use and re-use would be enhanced by identification of the intellectual content type of materials. A semantic registry that has the identifier of a scholarly content type as its primary key and that records various properties of the content type would support this. To facilitate syndicating, aggregating, post-processing and multi-purposing magazine, news, catalog, book, and mainstream journal content, the PRISM effort [25] has created such a vocabulary, but for materials typically used in a scholarly context it is lacking, making the semantics registry probably more critical to pursue than the format registry for which the MIME types can serve as a pragmatic stand-in. Semantic identifiers would be used for the semantic property available at the entity level of surrogates. Returning to the chemical search engine scenario, appropriate semantic identification of an entity would allow an agent to recognize it as a machine readable chemical formula, and thus choose to ingest the associated datastreams, the format of which can also be precisely described.

7 Experiments

To test the ideas presented above, we created obtain, harvest and put services to disseminate and ingest surrogates from and to several different repository architectures: Fedora, aDORe, DSpace and arXiv. We then used these interfaces to support the assembly of a number of articles from different repositories into a new issue of a hypothetical overlay journal. Instead of relying just on the user interfaces of the participating repositories, we also created a resource-centric search service using the harvesting infrastructure provided by the harvest interfaces. To further enhance the demonstration, we combined these techniques with Live Clipboard [35] technology to allow surrogates to be moved among repositories via the usual *drag-and-drop* metaphor. We first describe the two parts of this experiment, and then discuss implementation issues and experiences.

7.1 Harvesting journal articles to produce a resource-centric search service

A number of projects have attempted to use OAI-PMH harvested metadata for the creation of resource-centric or full-text discovery services. The principal problem is that resources cannot be unambiguously located from the simple Dublin Core metadata exposed by most OAI compliant repositories. This issue was discussed in detail in [17], where the use of complex object formats was proposed as a solution. While skeletal compared with formats such as METS and MPEG-21 DIDL, the surrogates proposed here also meet the requirements for resource harvesting. We implemented a search service based on the Nutch [36] crawler and search service. Instead of simply doing a web crawl, surrogates were harvested from the participating repositories using the harvest interface. Each surrogate was then introspected upon to select only those with the semantics "journal-article" (we agreed on a small ontology for these experiments). All the appropriate surrogates were examined to extract format and location information to dereference the datastreams. The datastreams were then fetched and indexed while retaining their association with the surrogate. This process is illustrated in Fig. 8.

In addition to the usual links back to the source repository and content excerpt, the search results display was augmented with a Live Clipboard icon allowing the surrogate to be copied into the copy/paste buffer on the user's computer, and thus easily passed to other applications as described below.

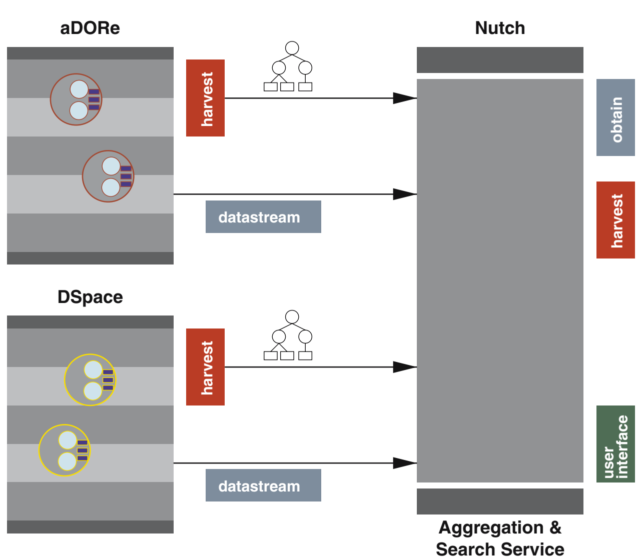

Fig. 8 Use of Nutch to create a resource-centric search service over repositories supporting the Pathways harvest interface

7.2 Creation of a new issue of an overlay journal

The scenario we have referred to most frequently is the composition of a new issue of an overlay journal from articles in different repositories. When combined with the search service just described, this scenario demonstrates the use of all three repository interfaces in a realistic scholarly value-chain. This scenario revolves around the editor of the overlay journal, "Ed". The key data flows as Ed interacts with one source repository are shown in Fig. 9, and the complete sequence of actions required to create the new overlay journal issue are described below.

1. **Select:** Ed applies whatever selection and review policies the overlay journal uses to decide which articles should be included in the new issue. Ed selects three articles; one each from arXiv, from an aDORe based repository, and from a DSpace repository.
2. **Obtain:** Consider first the selection of an article from arXiv. Ed navigates to the normal splash page for this article in whichever way is convenient, perhaps from Google, or from arXiv's own interface. The splash page not only displays the usual metadata, links to associate resources and links to the full-text, but also a Live Clipboard icon as shown in Fig. 10. By clicking on this icon, the Live Clipboard JavaScript uses arXiv's obtain interface to get a surrogate for the article which is stored in the copy/paste buffer of Ed's computer.
3. **Compose:** Ed then goes to the editorial web-interface for the overlay journal and pastes the surrogate via the Live Clipboard JavaScript on that page. Behind the scenes, the surrogate is put into the Fedora repository hosting the journal. Here it is a matter of local policy whether the ingest mechanism simply stores the surrogate with references to included entities and datastreams, or whether these are dereferenced and also ingested. For this demonstration we chose to ingest only the structural information — the entities — which simulates a "pure" overlay which simply links to articles in trusted repositories (perhaps with cyptographic signatures to guarantee that the original has not been altered). It would also be possible for the ingest system to implement a deep-copy and duplicate all the datastreams of a digital object. Note that in a real system Ed would have to authenticate with the repository for the overlay journal in order to be granted the privileges to put new content into the journal, presumably any attempt to put content by a non-authenticated and or unprivileged user would be denied.
4. **Complete composition:** A similar process is repeated for articles from DSpace and from aDORe. Here Ed uses the search service described in 7.1. The search results show a Live Clipboard icon with each result. By click-

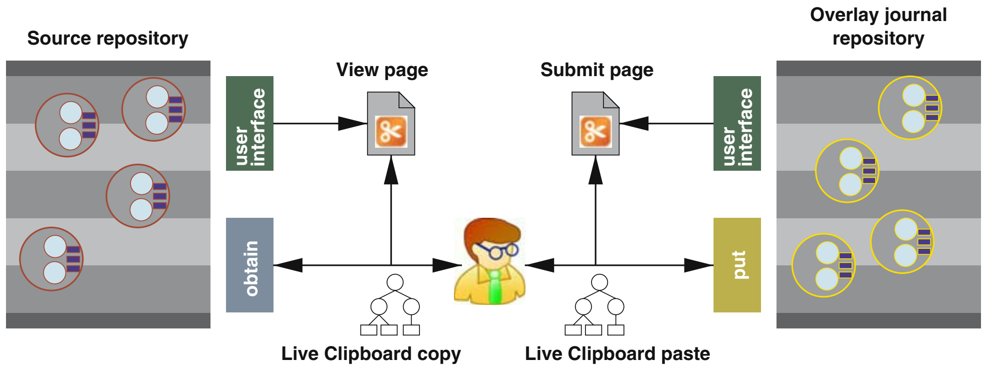

Fig. 9 Addition of an article from a participating repository to an overlay journal using Live Clipboard technology to copy a surrogate

ing on this icon, the Live Clipboard JavaScript uses the search service's obtain interface to get a cached surrogate for the article which is stored in the copy/paste buffer of Ed's computer. The overlay journal issue then has three articles queued.

5. **Submit:** When Ed is happy that all articles for the new issue are ready, the issue can be created as an entity in its own right by clicking the "Submit Issue" button. When this is complete, a surrogate for the new issue is available from the obtain and harvest interfaces of the overlay journal repository and may be used by all the same services that interoperate with the underlying repositories.
6. **Visualize:** To illustrate how other services can work within this framework, and to allow easy visualization of surrogates, we created an additional OpenURL-based service to visualize the surrogate graph using WebDot [22]. Example output for an arXiv article containing an additional data datastream is shown in Fig. 11. The OpenURL request simply includes the providerInfo of the surrogate which is enough to enable the surrogate to be obtained, rendered as an image of a graph with links to sub-entities, datastreams and registry entries for format and semantic URIs.

Though only a demonstration, the process of compiling a new issue for an overlay journal described above uses many interoperability features provided by the Pathways framework which are simply not available in existing systems. By implementing this over several of the most popular repository technologies we have demonstrated that this technology could readily be deployed.

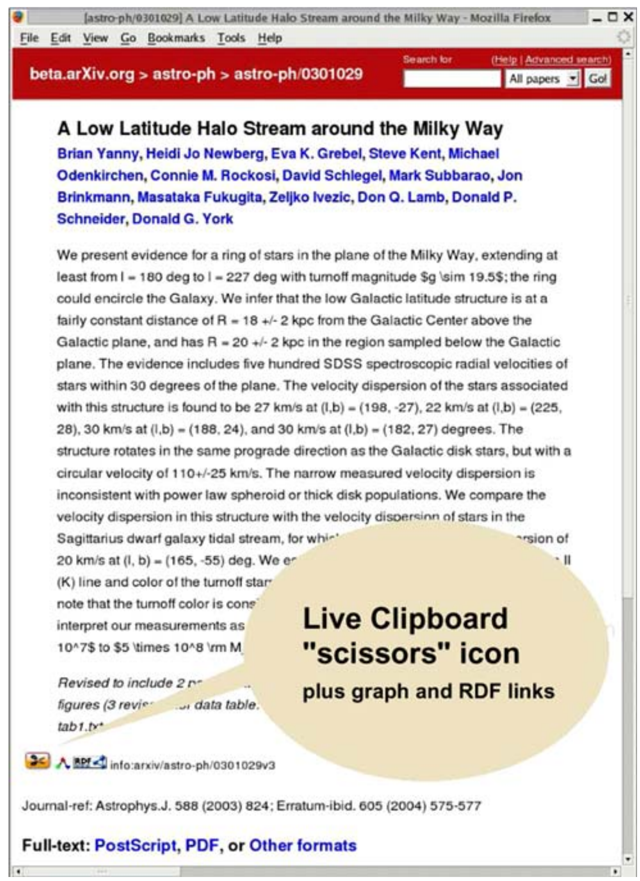

Fig. 10 Screenshot of the arXiv wrapper page augmented with Live Clipboard links

7.3 Implementation of the obtain and harvest services

The obtain interface is the simplest service and was the first that we implemented for each repository. By choosing to base it on OpenURL we were able to leverage existing OpenURL implementations for some of the repositories, simply adding another service identifier (svc_id) for the obtain service.

In implementing and obtain interface, one must work with the native data model of the underlying repository. The key decisions arise in translation of digital objects from their underlying representations to the Pathways Core model. Since the Pathways Core model is flexible, this can be done in different ways, depending on how much visibility Versus encapsulation of component parts is desired. Parts that are

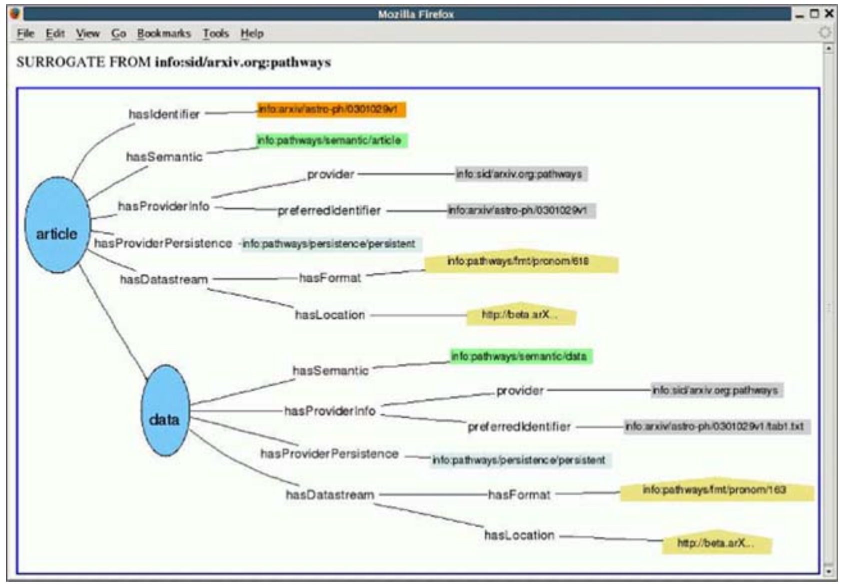

Fig. 11 Screenshot of the graph visualization showing an article from arXiv that contains a PDF file and a dataset

made available for re-use should be modeled as entities with associated providerInfo.

All of the repositories used in this experiment already supported the OAI-PMH so the implementation of the harvest interface was simply a matter of adding another "metadata" format for the Pathways Core surrogate to the existing OAI-PMH interfaces. It is a requirement of the OAI-PMH that all metadata formats be expressed in XML according to an XML Schema. Thus, we created surrogates using RDF/XML according to the W3C RDF/XML Schema [9]. Additionally, we agreed that, for convenience, all repositories would use a common metadataPrefix=pwc.rdf for this metadata format.

7.4 Implementation of the put service

While the put interface is agnostic to the particulars of underlying repository technology and models, any concrete implementation of a put service must be attentive to the specific capabilities and limitations of the underlying repository architecture. There are many questions that arise pertaining to how a put service interprets a surrogate and the assumptions the service makes in interacting with a particular underlying repository.

To support the overlay journal experiment, a put service was developed to interact with a Fedora repository. As no existing protocol already provided the required functionality, a new REST-based service for Fedora was created for our experiments. The flexibility of Fedora made it well suited for accepting and ingesting both simple surrogates (single entity) and complex surrogates (a graph of entities). However, this flexibility provoked the realization that there are a number issues to be considered in implementing an effective put service, which we detail in the following sections.

7.4.1 Identifiers and lineage

The put interface, itself, imposes no requirement on a receiving repository in terms of how it should deal with identifiers. However, whether the repository assigns new identifiers for surrogates ingested or not, it is important that there is a way to later determine that an entity is a new instance of an existing entity. The means that the providerInfo of the original surrogate should be retained in the hasLineage property of the new entity as described in Sec. 4.1.4. Thus the providerInfo provides the basis for "a chain of lineage" across multiple distributed repositories where each repository represents the same entity or entities in different contexts.

7.4.2 Ingesting hierarchies or networks of objects

There are many cases when a put service will receive a surrogate that models a hierarchy or graph of related entities. This presents a challenge in terms of determining an appropriate ingest policy for how surrogates will be processed, and what kinds of digital objects will ultimately be created in a receiving repository.

When surrogates contain a hierarchy of entities, some assumptions must be made as to the nature of the relationships of the entities in the hierarchy. Do parent-child relationships of a hierarchy imply a part-whole composition? Is the presence or absence of each part essential to the integrity of the whole? Alternatively, is the hierarchy to be interpreted as a looser containment relationship, where the integrity of the whole is not compromised if its parts are disassociated?

In our experiment, the put service assumed that any entity within a surrogate that contained providerInfo should be managed as its own digital object within the target Fedora repository. Thus, in the case of the journal overlay example, all journal, issue, and article entities were to be represented as separate Fedora digital objects with appropriate relationships asserted among them. Furthermore, any sub-entities of article entities with providerInfo were also represented as digital objects in their own right. In the experiments, there were article entities that were comprised of both a document and datasets, and each of these was represented as a separate digital object.

The end result of creating a journal issue via the put interface, was the creation of a graph of related digital objects in the target Fedora repository. From a management perspective, this modular and atomic arrangement can enable flexible management of objects. For example, it is easier to discover and do something with all dataset objects than it would be to find all types of objects that may encapsulate datasets. From an access standpoint, each component is registered as a digital object with its own public identifier and is available for re-use. This approach facilitated the ability to obtain the entire journal, or any sub-part down the hierarchy via the obtain service is a simple and generic manner. It was not necessary to create special services to discover and extract entities that were encapsulated within other objects. On the other hand, additional processes would be required to implement facilities that depend on handling all the constituent parts of a given digital object.

8 Future work

Much of the intellectual effort in this work has been to pare the Pathways Core data model down to its essential components. Having successfully carried out some initial experiments we intend to explore other scenarios, including those described in the introduction, to see where additional richness is required. We envision extensions or refinements to relationships in the model. One example is the notion of entity containment to include more specific semantics: distinction of part/whole vs. alternative or auxiliary; indication of equivalence; and the notion of ordering or not. In the overlay journal example, how and where should the order of articles be expressed in a surrogate for the journal issue?

The hooks from URI expression of semantic information and format identifiers open up tantalizing ties to current service matching work, the semantic web, and to ontology-based reasoning. It seems likely that PRONOM and GDFR format registries will coexist and be used by different segments of the scholarly community. How can we use ontologies in systems that will "understand" the equivalences and differences between these specifications? How can notions of generalization be applied to semantic information created in different contexts, and how can we service match on compositional semantics? For example, if an object contains a set of JPEG-2000 entities, should it be treated as a scanned book or a photo album?

If we imagine a landscape of widespread re-use of digital objects, there will undoubtedly be many copies and versions in different repositories and this provokes a number of questions. When designing a put interface, how can one understand if a surrogate duplicates an existing digital object? When a duplicate is discovered as part of a put request, should the current object be replaced? Or should multiple versions of the digital object be managed? These are repository-specific decisions, but whatever is decided may have significant implications in collaborative scholarly workflows. How should surrogates be validated?

The experiments described here have been performed over repositories that have primarily document content. The Pathways framework was conceived with a rich environment of documents, data and other media-types in mind. Future work will involve collaboration with other repositories that include significant data-repositories and other repository architectures.

9 Conclusions

Our experiments successfully demonstrated the ability to move surrogates of digital objects among repositories, and to re-use them in new contexts. The proposed interoperability framework allowed us to show how the basic workflow necessary to create a new issue of an overlay journal could be supported across heterogeneous repositories. The simplicity and generality of the Pathways Core data model allowed its use to create surrogates for digital objects held in Fedora, aDORe, DSpace and arXiv repositories, each with significantly different internal data models and architectures. Furthermore, by leveraging existing implementations of OpenURL resolvers and OAI-PMH interfaces for the repositories, it was remarkably easy to provide the dissemination (obtain) and harvest interfaces necessary for each repository to participate. The ingest (put) interface was implemented only for a Fedora repository and involved considerably more design decisions, many of which require further investigation to determine best practices.

These results are serving as the basis for further experiments with even more heterogeneous repository architectures, to include data repositories in particular. These experiments will implement other important value chains that are necessary to move toward the goal of the creation of a global scholarly communication system.

Acknowledgements We thank Rob Tansley for implementing the Pathways Core data model and associated dissemination services in DSpace; Chris Wilper for work on Fedora, including the put interface; and Lyudmila Balakireva for work on the WebDot service, and Zhiwu Xie for work on the the Live Clipboard infrastructure. This work was supported by NSF award number IIS-0430906 (Pathways).

References

1. Abrams, S.L., Seaman, D.: Towards a global digital format registry. In 69th IFLA (International Federation of Library Associations and Institutions) General Conference and Council, Berlin, Germany, August 2003. http://www.ifla.org/IV/ifla69/papers/128e-Abrams_Seaman.pdf [Accessed 29 June 2006]
2. Amazon S3 - Simple Storage Service. http://aws.amazon.com/s3 [Accessed 28 June 2006]
3. The Atom Publishing Protocol. http://www.ietf.org/internet-drafts/draft-ietf-atompub-protocol-09.txt [Accessed 30 June 2006]
4. RDF/XML Syntax Specification (Revised), W3C Recommendation, February 2004. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ [Accessed 28 June 2006]
5. Bekaert, J.: Standards-Based Interfaces for Harvesting and Obtaining Assets from Digital Repositories. PhD thesis, Ghent University, Belgium, 2006. http://hdl.handle.net/1854/4833
6. Bekaert, J., Van de Sompel, H.: A Standards-based Solution for the Accurate Transfer of Digital Assets. D-Lib Magazine, 11(6), June 2005. http://www.dlib.org/dlib/june05/bekaert/06bekaert.html
7. Bekaert, J., Hochstenbach, P., Balakireva, L., Van de Sompel, H.: Using MPEG-21 DIP and NISO OpenURL for the Dynamic Dissemination of Complex Digital Objects in the Los Alamos National Laboratory Digital Library. D-Lib Magazine, 10(2), February 2004. http://www.dlib.org/dlib/february04/bekaert/02bekaert.html
8. Bekaert, J., Hochstenbach, P., Van de Sompel, H.: Using MPEG-21 DIDL to Represent Complex Digital Objects in the Los Alamos National Laboratory Digital Library. D-Lib Magazine, 9(11), 2005. http://www.dlib.org/dlib/november03/bekaert/11bekaert.html
9. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, February 2004. http://www.w3.org/TR/rdf-schema/ [Accessed 29 June 2006]
10. The China Digital Museum Project: A DSpace federation. http://wiki.dspace.org/ChinaDigitalMuseumProject [Accessed 28 June 2006]
11. CrossRef.org: DOIs for Research Content. http://www.crossref.org/ [Accessed 28 June 2006]
12. Darlington, J.: PRONOM – A Practical Online Compendium of File Formats. RLG DigiNews, 7(5), October 2003. http://www.rlg.org/legacy/preserv/diginews/v7_n5_feature2.html
13. Van de Sompel, H.: Roadblocks, June 2003. http://www.sis.pitt.edu/∼dlwkshop/paper_sompel.html [Accessed 27 June 2006]
14. Van de Sompel, H., Beit-Arie, O.: Open Linking in the Scholarly Information Environment Using the OpenURL Framework. D-Lib Magazine, 7(3), March 2001. http://www.dlib.org/dlib/march01/vandesompel/03vandesompel. html
15. Van de Sompel, H., Bekaert, J., Liu, X., Balakireva, L., Schwander, T.: aDORe: A Modular, Standards-Based Digital Object Repository. The Computer Journal, 48(5):514–535, 2005. doi:10.1093/comjnl/bxh114
16. Van de Sompel, H., Lagoze, C., Bekaert, J., Liu, X., Payette, S., Warner, S.: An Interoperable Fabric for Scholarly Value Chains. D-Lib Magazine, 12(10), 2006, doi:10.145/october2006-vandesompel. http://dx.doi.org/10.145/october2006-vandesompel
17. Van de Sompel, H., Nelson, M.L., Lagoze, C., Warner, S.: Resource Harvesting within the OAI-PMH Framework. D-Lib Magazine, 10(12), 2004. doi:10.1045/december2004-vandesompel. http://www.dlib.org/dlib/december04/vandesompel/12vandesompel.htm
18. Van de Sompel, H., Payette, S., Erickson, J., Lagoze, C., Warner, S.: Rethinking scholarly communication: Building the system that scholars deserve. D-Lib Magazine, 10(9), 2004. doi:10.1045/september2004-vandesompel http://www.dlib.org/dlib/september04/vandesompel/09vandesompel.html
19. DLF Aquifer - Asset action experimentation. http://rama.grainger.uiuc.edu/assetactions/index.asp [Accessed 28 June 2006]
20. Information and documentation - The Dublin Core metadata element set, ISO (ISO 15836) and ANSI/NISO (Z39.85-2001), 2001. http://www.niso.org/standards/resources/Z39-85.pdf [Accessed 29 June 2006]
21. EduSource Communication Layer (ECL). http://ecl.iat.sfu.ca/#LORNET [Accessed 28 June 2006]
22. Ellson, J.: WebDot Home Page. http://www.site.uottawa.ca:4333/webdot/ [Accessed 28 June 2006]
23. Hitchcock, S., Bergmark, D., Brody, T., Gutteridge, C., Carr, L., Hall, W., Lagoze, C., Harnad, S.: Open citation linking: The way forward. D-Lib Magazine, 8(10), October 2002. http://www.dlib.org/dlib/october02/hitchcock/10hitchcock.html
24. Hunter, J., Choudhury, S.: A Semi-Automated Digital Preservation System based on Semantic Web Services. In Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital libraries (JCDL2004), pp. 269–278, Tucson, Arizona, USA, June 2004. http://doi.acm.org/10.1145/996350.996415
25. IDEAlliance and PRISM Working Group. PRISM: Publishing Requirements for Industry Standard Metadata, Version 1.2h, September 2003. http://www.prismstandard.org/PAM_1.0/PRISM_1.2h.pdf [Accessed 29 June 2006]
26. IMSCP: The IMS Content Packaging Specification. http://www.imsproject. org/content/packaging/ [Accessed 28 June 2006]
27. Isaac Newton to Robert Hooke in a letter dated 5th February 1676. Written as "If have seen further it is by standing on ye sholders[sic] of Giants.". Quote reproduced in Never at Rest: A Biography of Isaac Newton, Richard S. Westfall, Cambridge, 1980, p 274. Amusingly, many forms of this phrase are (mis)attributed to Newton. The phrase draws on the much older metaphor of "dwarfs standing on the shoulders of giants" and hence being able to see further by virtue of their vantage point rather than stature
28. Jacobs, I., Walsh, N.(eds): Architecture of the World Wide Web, Volume One, December 2004. http://www.w3.org/TR/2004/REC-webarch-20041215/ [Accessed 29 June 2006]
29. Jerez, H., Manepalli, G., Blanchi, C., Lannom, L.W.: ADL-R: The First Instance of a CORDRA Registry. D-Lib Magazine, 12(2), February 2006. http://www.dlib.org/dlib/february06/jerez/02jerez.html
30. JSR 170: Content Repository for JavaTM technology API. http://www.jcp.org/en/jsr/detail?id=170 [Accessed 28 June 2006]
31. Kahn, R., Wilensky, R.: A framework for distributed digital object services. International Journal on Digital Libraries, 6(2), April 2006. doi:10.1007/s00799-005-0128-x. Original 1995 report at: http://www.cnri.reston.va.us/cstr/arch/k-w.html
32. Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, February 2004. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ [Accessed 28 June 2006]
33. Kraan, W., Mason, J.: Issues in Federating Repositories: A Report on the First International CORDRA Workshop. D-Lib Magazine, 11(3), March 2005. http://www.dlib.org/dlib/march05/kraan/03kraan.html
34. Lagoze, C., Van de Sompel, H., Nelson, M., Warner, S.: The Open Archives Initiative Protocol for Metadata Harvesting, version 2.0, June 2002. http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm [Accessed 29 June 2006]
35. Live Clipboard Technical Introduction. http://spaces.msn.com/editorial/rayozzie/demo/liveclip/liveclipsample/techPreview.html [Accessed 2006-06-25]
36. Nutch: Open-source web search software built on lucence and java. http://lucene.apache.org/nutch/ [Accessed 29 June 2006]
37. The Library of Congress. METS: Metadata Encoding and Transmission Standard. http://www.loc.gov/standards/mets/

38. IFLA Study Group on the Functional Requirements for Bibliographic Records. Functional Requirements for Bibliographic Records, UBCIM Publications, New Series, 1998. http://www.ifla.org/VII/s13/frbr/frbr.pdf
39. Reference Model for an Open Archival Information System, CCSDS Secretariat, NASA, July 2002 http://ssdoo.gsfc.nasa.gov/nost/wwwclassic/documents/pdf/CCSDS-650.0-B-1.pdf
40. Open Service Interface Definitions (OSIDs). http://www.okiproject.org/ [Accessed 28 June 2006]
41. SRW/U Record Update service. http://srw.cheshire3.org/docs/update/ [Accessed 28 June 2006]
42. Atkins, D., et al.: National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure, Revolutionizing Science and Engineering through Cyber-infrastructure, 2003. http://www.communitytechnology.org/nsf_ci_report/
43. Waters, D.J.: Cyberinfrastructure and the Humanities. http://www.cni.org/tfms/2003b.fall/handouts/Fall2003Handouts/H- Watersplenary.doc [Accessed 27 June 2006]
44. WebDAV: Web-based Distributed Authoring and Versioning. http://www.graphviz.org/webdot [Accessed 28 June 2006]
45. XFDU: XML Formatted Data Unit. http://sindbad.gsfc.nasa.gov/xfdu/ [Accessed 29 June 2006]
46. ANSI/NISO Z39.88 - 2004 The OpenURL Framework for Context-Sensitive Services, 2004. http://www.niso.org/standards/-resources/ Z39_88_final_ANSIpending.pdf [Accessed 28 June 2006]