Augmenting interoperability across scholarly repositories

A meeting sponsored and supported by Microsoft, the Andrew W. Mellon Foundation, the Coalition for Networked Information, the Digital Library Federation, and the Joint Information Systems Committee.

April 20 – 21, 2006, New York, NY

prepared by Jeroen Bekaert & Herbert Van de Sompel

Participants at the meeting are Stephen Abrams (Harvard University), Jeroen Bekaert (Ghent University & Los Alamos National Laboratory), Peter Brantley (University of California), Tim Brody (University of Southampton), Rachel Bruce (JISC), Les Carr (University of Southampton), Lorcan Dempsey (OCLC Research), Tim DiLauro (John Hopkins University), Jeremy Frumkin (Oregon State University), Lee Giles (Pennsylvania State University), Jim Gray (Microsoft), Tony Hammond (Nature Publishing Group), Rachel Heery (UKOLN), Tony Hey (Microsoft), Randy Hinrichs (Microsoft), Thomas Krichel (Long Island University), Carl Lagoze (Cornell University), Cliff Lynch (CNI), Sandy Payette (Cornell University), Jerry Persons (Stanford University), Andy Powell (Eduserv Foundation), David Seaman (DLF), MacKenzie Smith (MIT), Rob Tansley (Hewlett-Packard), Herbert Van de Sompel (Los Alamos National Laboratory), Simeon Warner (Cornell University), Stuart Weibel (OCLC Research & University of Washington), Bill Ying (ARTstor) and Jeff Young (OCLC Research).

1. Context

Under guidance of the Andrew W. Mellon Foundation, the Coalition for Networked Information (CNI), the digital Library Federation (DLF), the Joint Information Systems Committee (JISC) and Microsoft, a meeting was held aimed at identifying concrete steps that could be taken to augment interoperability across heterogeneous scholarly repositories. The specific goal of the meeting was to try and reach a common understanding regarding a data model and a limited set of core, protocol-based repository interfaces that would allow services and downstream applications to interact with heterogeneous repositories in a consistent manner. Such repository interfaces include interfaces that support locating, identifying, harvesting, obtaining and depositing compound digital objects.

The meeting was attended by repository software representatives from such repository systems as DSpace, EPrints, and Fedora; content repository representatives from such organizations and companies as ARTStore, arXiv, Citeseer, Harvard Open Content, Nature Publishing; and technology advisors from such projects as aDORe, Pathways, Aquifer, Eduserv Foundation, NSDL and the DSpace Chinese Federation Project.

This is a brief report outlining this meeting. The agenda and presentations of the meeting are available at http://msc.mellon.org/Meetings/Interop/.

2. Scope of the meeting

A wide range of communities are building repositories as a means to store and share rich collections of digital objects. However, within and between each community there is a different perspective on the design and management of digital repositories. Repository systems may serve different user communities, have different policies regarding what is required for deposit, what storage mechanisms to use, which identification mechanisms to employ, what strategies need to be taken for long-term preservation, and so forth. This focus on particular materials and application domains has led to a variety of parallel technical approaches used in the design and implementation of repository systems, and as a result, to a serious lack of cross-community and cross-repository interoperability. Because of this, the realization of a truly interconnected digital knowledge environment that would have these digital heterogeneous repositories at its core and that would allow to readily use and re-use digital objects (hosted by these repositories) in rich value chains, remains unfulfilled. The exploration of this problem space and the basic agreement on a technical 'interoperability' solution were the subject of this meeting: Can interoperability be achieved given diversity of mission and practice? The aim is to define minimal common practice.

In order to devise such an interoperability framework, one needs to understand scenarios that drive the problem space. This was the subject of an introductory presentation presented by Herbert Van de Sompel (see Herbert Van de Sompel's introductory slides). In general, two major types of cross-repository value-chains motivate the quest for augmented cross-repository interoperability:

- Richer cross-repository services, that is services that overlay multiple repositories (e.g. discovery services, virtual collections, and so forth).
- Cross-repository scholarly communication workflows (i.e. digital objects contained in digital repositories are the subjects of scholarly communication workflows, and are used and re-used in many contexts)

Examples of such value-chains include:

- The transfer of (parts of) digital objects from digital repositories to parties that provide discovery-oriented services over (parts of) the digital objects. A 'chemical search engine' could make machine-readable chemical structures contained in digital objects searchable. And digital objects could be obtained from a variety of distributed repositories.
- The transfer of digital objects from digital repositories to parties that provide value-added services over the digital objects. For example, an editor could collect articles from different institutional repositories and submit them to an overlay journal that adds some value to them (e.g. by supplying metadata, or reviews for the articles).
- The introduction of natively machine-readable and machine-actionable bibliographic citations.
- The re-use of datasets (from different repositories) as the basis for building a new dataset or for writing a publication. Again, the newly created digital object can be deposited in and be re-exposed by a digital repository.
- *"The mirroring of digital objects between repositories to guarantee the existence of backups" (Cliff Lynch).*
- *"The building of virtual collections of digital objects. A digital repository may present a coherent view of a collection that actually spans a multitude of digital repositories that hold responsibility for it. In such a scenario, the individual members of the virtual collection should not be materialized until needed. We should not replicate the digital objects in the (system representing the) virtual collection; instead we will need some form of redirection." (Cliff Lynch).*

Note that we can realize some of these scenarios already; yet only in idiosyncratic, that is repository-specific or manual ways. We also typically loose the connection between the source digital object as it resides in a digital repository, and a digital object that is created as the result of a value chain which has other digital object(s) at its origins.

3. Expectations of the meeting

The meeting focused on the appropriate level of interoperability required across heterogeneous repositories to enable scenarios such as the ones described in the above. More specifically (see also http://msc.mellon.org/Meetings/Interop/goals_and_topics):

- To agree on the nature and characteristics of a limited set of core, protocol-based repository interfaces (REST-full and/or SOAP-based Web services) that allow downstream applications to interact with heterogeneous repositories in an efficient and consistent manner.
- To compile a concrete list of action items aimed at fully specifying, validating and implementing such repository interfaces.
- To devise a timeline for the specification, validation and implementation of such repository interfaces.

As Cliff Lynch put it aptly: *"I am not expecting finished specifications, protocol practices or running implementations yet; rather some consensus on general interoperability approaches and some agreement on how to carry out specific pieces of these approaches. Experience tells us that this kind of work needs to be discussed in length and validated experimentally. I am hoping that we can leave at this meeting with a pathway that will take us to this sort of experiments in the coming months."*

4. What is a digital repository?

In order to explore this problem space, an understanding is required regarding the nature of digital repositories and the kind of compound materials they store. In the introductory presentation, Herbert Van de Sompel put forward the following working definition of a digital repository (see Herbert Van de Sompel's introductory slides):

A repository is a networked system that provides services pertaining to a collection of digital objects. Example repositories include: Institutional repositories, Publisher's repositories, Dataset repositories, Learning Object repositories, Cultural Heritage repositories, etc.

Also, as explained by Cliff Lynch: *"Currently, we do not really have a shared definition of a digital repository. We lack the array of technical specifications to test the conformance of a digital repository in an environment of interoperability. If I point to a given thing and ask you whether or not this is a digital repository, we would differ. The best thing we can say is that if it doesn't support the Open Archive Initiative Protocol for Metadata Harvesting, it is probably not a very well-behaved digital repository, assuming it is a repository. In this meeting, we will try to provide a number of these sort of technical/behavioral characteristics of digital repositories."*

5. What is a digital object?

The following working definition was put forward: A digital object is a data structure whose principal components are digital data and key-metadata. Digital data can be a datastream or a digital object, i.e. a digital object may have one or more other digital objects as nested components. Key-metadata must include an identifier for the digital object. Note that digital object as used here is similar to the Kahn/Wilensky digital object. (see Herbert Van de Sompel's introductory slides).

"Over the last couple of years, many areas in the scholarly domain are focusing on (compound or complex) digital objects: We have e-learning objects, art objects, museum objects, e-science datasets, chemical databases, e-books, journal issues, compound articles, and so forth. Each of these digital objects may consist of multiple datastreams, that, in some way, are related to each other. There is a need to compose, decompose and structure these digital objects. Some constituents of a digital object could be stored in one place; other parts could be stored in other places, and so on. Many scholarly areas are getting increasingly dependent on these compound digital objects." (Don Waters).

Comment on the need for new terminology?

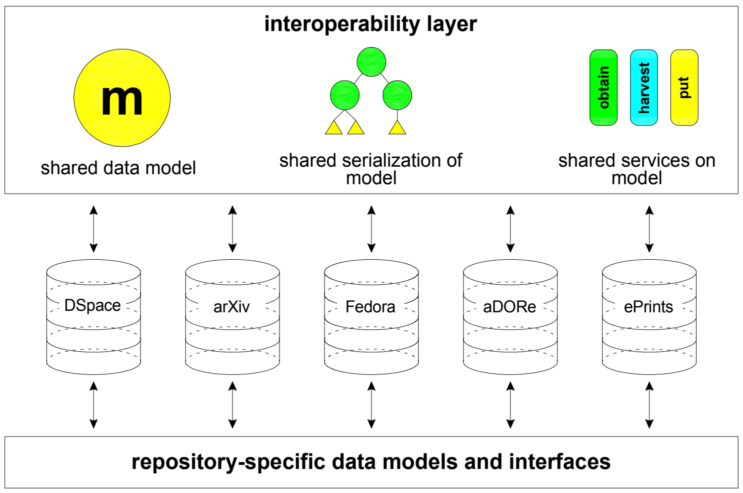
The term 'digital object' also raises several questions, because (see Andy Powell's slides on harvest functionality):

- Scholarly (and other) repositories will need to expose metadata about entities other than digital objects (e.g. physical people and abstract works). Therefore, it is not clear that 'digital object' refers comprehesively to all aspects of digital repositories that need to interoperate.
- The Architecture of the World Wide Web uses the terms 'resource' and 'representation', approximately where we are using digital object and surrogate. However, the data model for resources of the World Wide Web is less expressive than the data models that are currently used to model (compound) objects from the scholarly community. Would it be possible to reconcile these different concepts?

6. Devising a cross-repository interoperability layer

Based on inspiration gained from projects such as Pathways, aDORe, CORDRA, the Chinese DSpace Federation project, various JISC projects, and the OAI effort, the following path towards augmented interoperability is proposed (see Herbert Van de Sompel's introductory slides, the terminology list in Annex, and the figure below):

- Support for a **data model** for digital objects that is supported across repositories. A data model is an abstraction (or an extra level of indirection) for digital objects such that each digital object can be seen as an instance of the class defined by the data model. The data model intends to provide a common represention of digital objects in a set of heterogeneous digital repositories. Example data models include Pathways Core, the MPEG-21 digital Item Declaration Abstract Model, the OAIS Information Model, and so forth.
- Support for a **surrogate** that serializes (or actualizes) the digital object in accordance with the data model. In other words, a surrogate is a transmittable serialization or representation of a digital object that can be passed back and forth so we can do things with it. Possible serialization techniques include XML and RDF/XML.
- Support for a set of core services (and interfaces) that have to be supported by a well-behaved digital repository. In the proposed architecture, a distinction is made between:
    - An **obtain** interface: A repository interface that supports the request of services pertaining to individual digital objects. A distinction can be made between at least two interface levels. 1) An obtain interface from which a surrogate of an identified digital object can be obtained. This interface would need to be natively supported by each individual repository. 2) An obtain interface from which a list of repository-specific services pertaining to an identified digital object (and its constituent datastreams) can be obtained. This interface could be supported by individual repositories, but could be implemented as a repository overlay.
    - A **harvest** interface: A repository interface that exposes surrogates for incremental collecting/harvesting.
    - A **put** interface: A repository 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.



To stimulate discussions, ideas regarding interoperability were presented, and a demonstration of a prototype resulting from the Pathways project was shown. In what follows, a brief summary of these presentations is provided. Questions and comments resulting from the discussions following these presentations are summarized throughout the text.

Data model and surrogate

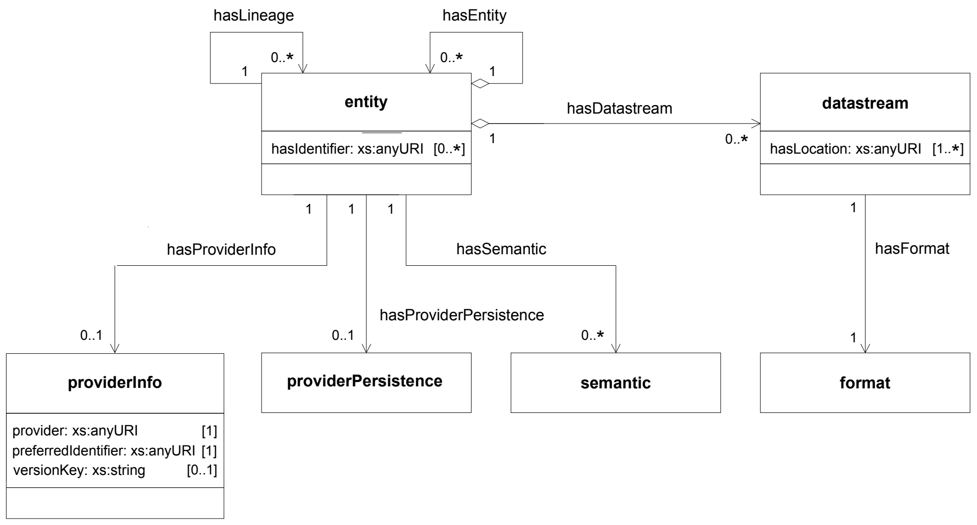
The notion of a data model is important as it introduces a higher level of abstraction (to talk about digital objects) that may persist over time, while serialization techniques may change over time as technologies evolve. The introduction of a higher level of abstraction also resonates well with the long-term perspective of scholarly repositories. Furthermore, it introduces a certain degree of flexibility as it allows for many serialization schemes (conformant with the data model) to exist in parallel.

A presentation by Carl Lagoze introduced the Patwhays Core data model. The Pathways Core data model is being proposed as an interoperable data model with a specific focus on applicability for cross-repository services and workflows (Bekaert et al, 2006). It allows one to convey properties and access points pertaining to a digital object in a uniform and repository-neutral manner. A Unified Modelling Language (UML) structure diagram of the Pathways Core data model is shown below.

At the heart of Pathways Core are the entity and datastream elements. entity elements model the **abstract** aspects of digital objects. entity elements are recursive and can model anything from a (part of a) digital object to a collection of digital objects. The core properties of entities are hasIdentifier, hasProviderInfo, hasLineage, hasProviderPersistence and hasSemantic. Each of these properties are explained and discussed in more detail in the succeeding subsections.

The datastream elements model the **concrete** aspects of a digital object, and can be thought of as aspects at the level of bitstreams. An entity may have any number of datastreams. Two properties of a datastream have been defined as part of the Pathways Core: hasLocation conveys a URI that can be resolved to yield a bitstream; and hasFormat conveys the digital format of the bitstream.

The Pathways Core data model can be actualized in a variety of ways, and an RDF syntax has been used in the demo that is presented at the meeting.



Comments on a data model and its long-term persistence.

"Why would we care about what happens in two or three hundred years from now? It imposes us with a responsibility that we currently cannot cope with. Maybe we should focus on what happens in the next twenty or thirty years." (Andy Powell)

"Even if you are thinking in the twenty year time frame, you still need your abstractions. You cannot tie everything to, for example, HTTP GET. Today, you have to support different types of identifiers already". (Herbert Van de Sompel)

6.1.1. Pathways Core tree structure

In Pathways Core, the tree structure of a (compound) digital object is represented using hasEntity and has-Datastream properties. Digital objects can be composed of one to zero entities and/or one to zero bit-streams. entity elements model the abstract aspects of digital objects and can be nested recursively, while datastream elements are the concrete bitstreams. This nesting capability allows for digital objects (or parts thereof) to contain other digital objects and allows for the association of properties pertaining to (or shared by) the underlying content.

This 'containment' concept stems from the Kahn/Wilensky Framework that refers to digital objects as data structures whose data consists of (sets of) bit-sequences and/or (sets of) digital objects. A digital object that contains other digital objects is said to be composite. A digital object that is not composite is said to be elemental.

This general model seems pretty uncontroversial and well-established, and there hardly was any debate about this at the meeting.

Comments on deep and shallow copies of a digital object

During the meeting, several discussions focused on the difference between 'surrogate transfer' and 'digital object transfer'.

As explained in Carl Lagoze's presentation on the data model, there are obviously applications where digital object transfer (instead of 'just' surrogate transfer) is very important. Such applications include mirroring applications, preservation applications, the transfer of digital objects to trusted repositories, etc. Yet, full asset transfer is only necessary for some applications. It is not a mandate of the full application realm that we are investigating here. In many cases, only parts of a digital object need to be transferred in order to enable an application. And, in a world of re-use, some application domains even forbid transfer of the actual digital object. As such, the data model should accommodate but not be limited to full asset transfer. By not committing to asset transfer, the following advantages are introduced:

- Avoid having to deal with Intellectual Property (IP) issues in the interoperability layer, and rather push IP issues to an application layer.
- Accommodate the notion of 'service-tuned' asset transfer. The service layer sits in the driver seat, and may request what it needs (shallow vs deep copy). This could be achieved using a parametrized obtain request, or by inspecting the semantic properties of the constituent parts of a digital object as expressed by its surrogate.
- Allow a world of live references rather than a world of static copies. In scholarly communication many digital objects are alive. A copy of a digital object can be retrieved at a certain point in time, but the copy is a surrogate of the digital object. This surrogate can be used as a means to get to the live constituent parts.

Comment on the IP associated with surrogates.

"I don't believe that surrogates can be free of IP. As soon as they get implemented someone will want to claim rights to them." (Andy Powell)

Comments on the inline provision of datastream versus the provision of pointers to datastreams.

"I think it needs to go both ways. If you pass on a surrogate and the source repository is going down, you obviously need datastreams in the surrogate. As long as the source is alive you can do call-back requests. Both scenarios need to be supported in the data model." (Sandy Payette)

"The support of a provision technique for datastreams is a property of the serialization syntax, not of the data model." (Simeon Warner)

"It may be important to note that the by value provision technique – that is the provision technique by which binary datastreams are base64-encoded before they are embedded in an XML-based wrapper – rapidly leads to memory problems at both the end of the repository and the end of the harvester. Though, this may be a temporary consideration that is typical of today's technology." (Herbert Van de Sompel)

Comments on pointers to datastreams

"If you have a surrogate – and let's presume this surrogate provides pointers to datastreams –, then you don't know what those pointers are really pointing at. For example, in a DSpace system, a pointer typically links to a file stored in the system. But Fedora has this notion of on-the-fly service transformation, where you can demand a datastream and feed it to a web service and back will come a dissemination of that datastream. For example, the pointer may point at a Word document that – through a service negotiation – comes out as a PDF document. In the Pathways data model, this is abstracted out. A pointer is defined as an access point." (Herbert Van de Sompel)

6.1.2. Decorating the Pathways Core tree structure

Other properties of the Pathways Core *data model* are about – what Jim Gray referred to as the process of – ***decorating the nodes of the tree structure*** with properties. An overview of the core properties as described by the presenters is given below. Several quotations from discussions are added as well.

1. Identifier of the digital object

In Pathways Core, the identifier of the digital object is conveyed using the **hasIdentifier** property. The property is neutral with regard to the type of identifier that is used for the identification of the digital object. This allows the data model to be implemented for a broad variety of repository systems, in which, for cultural, political, organizational and/or technical reasons, different choices are made regarding identification schemes. The decision as to whether or not (and when) a (revised) digital object should receive a new identifier or rather a version indicator is a matter of repository policy.

2. ProviderInfo: Obtaining the surrogate

In Pathways Core, the **providerInfo** element records the information that is needed to obtain a surrogate. This information is a triple consisting of 1) the identifier of the repository that exposes the surrogate 2) the preferredIdentifier of the digital object and 3) an optional version key. By appending providerInfo to an entity node, one allows a serialization (surrogate) of that node to be obtained and re-used in value-added services. It specifies the granularity of re-use of a digital object. The process of obtaining such a surrogate can be outlined as follows:

"*First, you use the identifier of the digital repository (that is the identifier of the provider) for a lookup into a service registry. That registry returns information (such as the location of the service interface) for different services available from that repository. One of these services is 'obtain': A service that allows you to request a surrogate of a digital object by using the preferredIdentifier of the digital object.*

Second, you can use the preferredIdentifier of the digital object (and the optional version key) to obtain a surrogate for the identified digital object from the obtain interface." (Herbert Van de Sompel)"

Comments

"I am rather uncomfortable with the notion of providerInfo and the concept of identity consisting of three parts. By doing so, I think you have complexified the notion of identity. Also, why do you think providerInfo is core?" (Stu Weibel)

"*ProviderInfo is the key information for obtaining a surrogate. The information to obtain a surrogate is declared within the surrogate itself. This shows a remarkable parallel with an HTTP response that declares its own HTTP address." (Herbert Van de Sompel).*

"Somewhere you will need a date in there." (Clifff Lynch).

3. Persistence of obtaining a surrogate using ProviderInfo

The **hasProviderPersistence** property attached to an entity expresses a level of commitment of the provider regarding returning a surrogate in response to future obtain request using the providerInfo that was attached to the entity. The commitment can vary dependent on repository policies, types of materials, and so forth. (See Carl Lagoze's slides on the data model).

This property generated much debate as indicated in the selected comments below. Cliff Lynch reconciled the different views as follows:

"The various relationships between surrogate, surrogate-persistence, persistence of underlying material, and so forth need to be carefully clarified. We need to understand what is policy related, and where the technical mechanisms fit in. It is also important to emphasize the unidealistic nature of current practice. We talk about immutability of articles, but in fact, today in practice, you have all kinds of publishers making tiny corrections to existing digital objects. Everyone has a different policy on how much tweaking and versioning is allowed. We cannot fix that through these mechanisms. But we should be aware that there is some area for judgement." (Cliff Lynch).

Comments on the immutability of surrogates

"*A surrogate cannot change. It is a serialization or a view of a digital object at a specific moment in time. Hence, by definition, surrogates are immutable." (Jim Gray/Carl Lagoze)*

"A surrogate that I obtain today is fixed. The surrogate sits here and does not change. However, by using the providerInfo contained in that surrogate, I can obtain a new surrogate (of the same digital object). Both surrogates can be different as the underlying digital object (and/or its datastreams) may evolve over time." (Herbert Van de Sompel)

Comments on the persistence of the digital object and its underlying content (e.g. transient datasets, etc.).

"I would argue that providerPersistence is a set of variable declarations about both the availability of the digital object and the immutability of the digital object. There is a slot in the surrogate that tells you: I as the provider of this digital object, assert that this digital object is immutable. The digital object will always be here for the rest of eternity." (Carl Lagoze)

"providerPersistence is key to indicate whether or not the content underneath the hood of the surrogate is stable or volatile thing. The providerPersistence conveys a reflection, an assertion of whatever it is." (Sandy Payette)

"There clearly are two different things: First, there is the surrogate itself and expressing whether or not you will be able to go back to the repository and obtain a new surrogate. Second, there are these assertions about the content itself. This is another issue, that should be expressed in parallel with the previous one, and so far, has not been addressed by the data model." (Herbert Van de Sompel)

4. Lineage: Capturing the surrogate workflow

The **hasLineage** property conveys an audit trail that captures the movement (or workflow) of a surrogate. The information contained in the lineage property is copied from providerInfo: It contains the information that is needed to obtain a surrogate of the digital object that has been used as input to the workflow process. By embedding lineage within the surrogate, the surrogate shows evidence of its workflow.

Comments on the granularity of Lineage

"At the technical level, the concept of lineage is very clear: You copy the information conveyed by the providerInfo element of the surrogate you received and paste that information in the lineage element of the surrogate you create" (Herbert Van de Sompel).

"The Pathways Core model focuses on two kinds of relationships: containment and lineage. Both are inherent to the Pathways Core Model and deserve to be separated out" (Sandy Payette).

"I think lineage is a superclass of a number of relationships that you may want to express. Currrently, the model allows one to make a 'hasLineage' assertion, but it would be nice to have a property of lineage to express the workflow at a somewhat finer level of granularity (e.g. isCopyFrom, isDerivationFrom, etc.)". (Andy Powell).

"Such a need for finer level of granularity may apply for containment as well" (Les Carr).

Comments on the privileged treatment of Lineage

"Lineage is a concept that exists within the space of the data model: It lives in this space, and hence, is best represented within that space." (Simeon Warner).

"It is not a matter of privileging lineage. It is a matter of recognizing that – within this data model – new levels of interoperability can emerge; and in order for these new levels to work properly, you have to keep some metadata about it." (Les Carr).

"Lineage is not 'just' metadata: It is the intellectual history of where something came from and how it came to you and what has been done to it along the way. Lineage is the machine-readable documentation of a mechanical process, namely the process of moving surrogates. It is automatically recorded as the by-product of performing these mechanical processes" (Cliff Lynch).

Comments on the relationship between Lineage and datastreams

"Lineage captures the workflow of a surrogate and not of the content or datastreams. This may cause problems because of the variable ability to inline datastreams in a surrogate. You may want to capture service operation that have been applied upon the datastreams as well. So, you may wind up also putting this property on datastreams." (Cliff Lynch).

5. Formats and semantics of the digital object and its constituents

In pathways Core, the **hasSemantic** property conveys the 'genre' of a digital object or constituent thereof. The **hasFormat** property conveys the information needed to decode or understand the bit structure of a datastream. Both genre and format facilitate the association of rich services. A good example of MIME based service matching is Jane Hunter's PANIC work.

Formats are typically expressed using MIME types. However, it should be noted that MIME types are limited in scope. For example, the MIME type 'application/pdf' does not specify what version of PDF the file is, and different PDF versions have very different features and behaviors. Hence, presenters recommended that both genre and format are expressed using globally unique identifiers. And each identifier should correspond with an entry in a genre or format registry (see also). Examples of format registries are PRONOM and the Global digital Format Registry.

Comments on Abstract versus Concrete

"One does not want to lock the data model to describing those things of the digital object that can be streamed. You want the model to describe the more abstract structural aspects of the digital object as well. For example, a bibliography is an abstract concept that is streamable in a number of different formats. The bibliography can be represented in BibTeX, in endnote, in PDF, and so on. However, the bibliography itself is at a somewhat higher level than the MIME. These higher level aspect can be labelled using the hasSemantic property" (Carl Lagoze)

Comments on the optional provision of Semantics

"The semantic property is service enabling. The semantic is also optional. If a surrogate has no semantics attached to it, the surrogate is low compliance" (Sandy Payette)

"Doesn't that lead you to a point where it is impossible to replicate digital objects from one repository to another; because – without semantics – you have no comprehensive representation of the abstract digital object. All you have is a series of concretizations of it in different formats, which may vary from repository to repository." (Cliff Lynch)

"Arguably it is impossible to copy a digital object from Fedora to DSpace because they have completely different views on what a digital object is. You can copy a representation of a digital object, but you can't copy the object itself. The proposed pathways data model may provide a thin layer of homogeneity that allows you to copy views/representations of the object that are compliant with the model. (Carl Lagoze)

Core Services

This section lists the core services that have to be supported by a well-behaved *digital repository*. A distinction is made between *harvest*, *obtain* and *put* which have been presented by Andy Powell, Herbert Van de Sompel and Rachel Heery, respectively.

6.1.3. Harvest interface

A Harvest interface is a repository interface that exposes surrogates for incremental collecting/harvesting. Of crucial importance is the concept of so-called selective harvesting that allows downstream applications to harvest only surrogates for those assets that were created or modified within a specified date range. The following harvesting scenarios were presented (See Andy Powell's slides on the harvest functionality):

- Aggregator. A service that gathers surrogates from multiple repositories and exposes them for harvesting by other applications/services
- Abstracting and Indexing. An application that gathers surrogates from multiple scholarly repositories and requests the associated datastreams and bibliographic metadata using the obtain interface. The combination of this information is used to index, interlink and rank the available digital objects
- Archival Storage. A service that uses the harvest interface and obtain interface to regularly migrate copies of digital objects to an archival store with the aim of long-term preservation

A well-established framework for harvesting materials is the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). While typically the OAI-PMH is used to harvest XML-based descriptive metadata records, like Dublin Core or MARCXML, under the augmented model, the OAI-PMH would be employed as a means to harvest surrogates. Obviously, these surrogates would need to be conformant with the afore-described data model.

The OAI-PMH has received a high level of buy-in from the digital library community and has established itself as an important solution for exchanging descriptive metadata among a large and varied group of digital repositories. It is thus intriguing to consider using existing installations of the OAI-PMH (that expose descriptive metadata) as a leverage to promote the widespread adoption of content harvesting interfaces.

When using the OAI-PMH as a protocol to harvest surrogates, the following observations can be made (See also Andy Powell's slides on the harvest functionality):

- There is a potential requirement for harvesting requests based on the digital object identifier, that is the OAI-PMH resource identifier, whereas current OAI-PMH requests are made in terms of the OAI-PMH item identifier.
- Similarly, datestamps in OAI-PMH pertain to the metadata record rather than the digital object. Hence, some work needs to be done on scoping the meaning of an OAI-PMH datestamp in a world of compound digital objects. The various relationships between a datestamp and a surrogate, a datastream and a digital object need to be clarified.
- It should also be noted that the OAI-PMH is not specified in a persistent manner. The OAI-PMH specification is closely bound the HTTP protocol and the XML syntax for transporting and serializing the harvested records. The OAI-PMH might benefit from a more abstract definition.

Other harvesting technologies are the Really Simple Syndication 2.0 (RSS) – in combination with content negotiation plugins, such as Microsoft's Simple Sharing Extensions – and the Atom Publishing Protocol. Especially the latter is gaining popularity.

A simplified view on harvest: ListIdentifiers & Obtain

"We may not need a harvest interface in the way that we understand it in terms of the OAI-PMH. The ListRecords verb appears to be redundant. A ListIdentifiers, probably combined with date scoping, followed by an Obtain request (per identifier) may handle what we need." (Andy Powell)

"This can indeed be considered an optimized view on harvest and somehow relates to performance, because we seem to introduce three steps (that is ListIdentifiers, obtaining the surrogate and resolving the datastream locations) instead of two (harvesting the surrogates and resolving the datastream locations)" (Herbert Van de Sompel)

6.1.4. Obtain interface

An obtain interface is a repository interface that supports the request of services pertaining to individual digital objects (including their component datastreams). A distinction can be made between at least two levels (See also Herbert Van de Sompel's slides on the obtain functionality). In the context of this meeting, the emphasis is on the first (and simple) level of conformance only.

- ***A simple level of conformance***: This level supports requesting a representation of an identified digital object. For example, requesting a pathways core surrogate of a digital object

  "Looking at it this way, you can see this obtain as a global mechanism to resolve an identifier into a surrogate. You can use the providerInfo – that contains the identifier of your repository – to do a lookup in the service registry. There you figure out where the obtain interface for that repository is located. You can use this information, plus the identifier and version key of the digital object to request a surrogate of that object" (Herbert Van de Sompel)

- ***A more advanced level of conformance***: This level supports requesting services pertaining to a specific digital object, including its datastreams. For example, requesting a certain datastream or request a PDF version of a MS Word datastream. This level is beyond the scope of this meeting.

The NISO OpenURL Framework for Context-Sensitive Services can be used as the basis for the obtain interface. The OpenURL Framework provides a means for defining a service environment, in which packages of information are transported over a network. These packages have an identifier of a referenced digital object at their core, and they are transported with the intent of obtaining context-sensitive services pertaining to the referenced object. To enable the recipients of these packages to deliver such context-sensitive services, each package describes the referenced asset itself, the type of service, the network context in which the asset is referenced, and the context in which the service request takes place. Two properties that are worth emphasizing are:

- The OpenURL Framework allows a repository or agent to include information about the context in which the obtain request took place. That may not be all that important for requesting the surrogate; though may prove to be essential when requesting services pertaining to the datastreams.
- The OpenURL Framework has been defined in an abstract (and hence, persistent) manner: A clear distinction has been made between abstract definitions of OpenURL concepts and their concrete implementation. It also supports identifiers from any namespace. There is no requirement to select specific identification technology.

Comments on parameterizing requests to obtain shallow or deep copies *(see also section 6.1.1)*

"One should be able to request shallow or deep copies of a digital object; One may not get it though…" (Sandy Payette)

"In a binary world, the options are twofold: either full or none. But perhaps in this context, there are intermediate solutions." (Cliff Lynch)

"It may be reasonable to define a lowest common denominator: For example, every repository must be able to expose a shallow surrogate as a lowest common denominator" (Cliff Lynch).

6.1.5. Put interface

A put interface is a repository 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. Possible scenarios are (See also Rachel Heery's slides on put functionality).

- The put interface enables users to populate repositories effectively. For example, an author 'saves' a report from a desktop authoring system to the institutional repository.
- The put interface enables repositories to exchange data in a predictable manner. For example, an institutional repository submits a learning object to a national learning object repository.
- Experimental data output from a spectrometer is 'saved as' a file and a file containing metadata on operational parameters is also generated. A data capture service is invoked and the files pertaining to the experiment are deposited, along with the necessary metadata, in the laboratory repository.

Many comments were made; two of which are quoted below. Cliff Lynch reconciled the different views as follows:

"On the put side I am hearing two things. The urgency about put is really about the population of primarily institutional repositories. It would be useful to build tools that assist in populating local institutional repositories. However, there is a second agenda. That is, it would be nice to be able to propagate this put functionality elsewhere, in particular for cross-repository scenarios. I do not feel that we have really begun exploring the pros and cons of this second agenda, other than the expressions of a couple of immediate objections about how it connects to local repository policies, authentication, and authorization. There is no consensus on these cross-repository scenarios yet.

The real point of pain right now, is the fact that we would like to expedite the population of local repositories. Hence, a set of interoperable tools that support the population of digital repositories is essential. It is also clear that the right name for this kind of functionality is not 'put'; but rather 'deposit request' or something similar. It is also clear that this functionality is not about putting things into a repository but rather about queuing things for an ingest process.

It also seems likely that while we might say that we really hope all repositories support the obtain and harvest interfaces, this queue-for-ingest interface will only be appropriate for a certain subclass of repositories." (Cliff Lynch)

Comments on put versus queue-for-ingest vs deposit request

"We need to make a clear distinction between the deposit authentication architecture on the one hand and the request for putting something on the other hand. 'Put' is just one step in a series of steps that need to be standardized. We should not overload this step." (Sandy Payette)

"You won't be able to standardize the complete deposit and ingest processes of a repository. These are policy dependent. But you can model the initial step that bootstraps the processes. The step in which you launch a request for ingest of a surrogate into the repository." (Tim DiLauro)

"Hence, I suggest we label this functionality a 'deposit request' instead of 'put'." (Andy Powell)

Comments on the leverage of put

"Put or queue-for-ingest has very little to do with cross-talk between repositories. It appears to be an application. We already have a huge leverage of obtain, and may consider implementing the put functionality as a local application, by using external out-of-band mechanisms" (Peter Brantley)

"It is rather a question of how much you want to solve in an interoperable way; how much you want to throw into the interoperability pot. By using out-of-band mechanisms, you are not going to break anything. It means that it will be solved in idiosyncratic ways. It will still be done. Things are getting into the internet as we speak." (Carl Lagoze)

Additional service interfaces

During the meeting, it was argued that the interoperbility framework may benefit from the introduction of a search interface and a publish-subscribe (PubSub) service. In general, it was felt by the Working Group that – while both sets of interface are essential – they should not be part of the core 'plumbing' and can be implemented as autonomous services that overlay one or more digital repositories and that are created through interaction with truly core repository interfaces for harvesting and obtaining. A few further observations are provided below.

6.1.6. Search interface

"In creating the Protocol for Metadata Harvesting, we faced vast evidence that doing search is very complicated; especially if you want to incorporate federations at arbitrary levels. In contrast with search, harvest is a fairly deterministic process and enables search services to be build on top of it." (Cliff Lynch)

"Search gets very complex in a world of complex objects. How do you define it, let alone specify it in an interoperable manner?" (Herbert Van de Sompel)

"Nonetheless, we may want to give some guidance on a standard search interface. Several repositories already have some kind of Z39.50 or SRU/W based interface. It appears to be quite reasonable to start a research project that is tasked with exploring any kind of practices for setting those up in a world of complex objects." (Cliff Lynch)

6.1.7. Subscribe interface

Publish-Subscribe (PubSub) is a **subject-based** and **event-driven** service that instantly notifies you when new content or digital objects are created that match your subscription.

“A PubSub service uses a controlled vocabulary to check whether articles have key words that match the subscription. The interpretation of the ‘matching’ is site dependent. The user is notified of new information that matches its subscription. This can, for example, be done using RSS or Atom” (Jim Gray)

Infrastructure components

Another presentation at the meeting was on registries and infrastructure (presented by Jeremy Frumkin). In this presentation, it was explained that the proposed interoperability framework would require a few supporting infrastructure components:

- There is an essential need for a **service registry** that facilitates locating the core interfaces (obtain, harvest and put) of the individual repositories that participate in the framework. A service registry has the identifier of the repository as its primary key and records the services and the network location of the service interfaces.
- There is a requirement for a **format registry** and **genre registry** recording values that can be used in the hasFormat and hasSemantic properties, respectively. The former has the identifier of a media format as its primary key and records various properties of the media format. The latter has the identifier of a semantic type (or genre) as its primary key and records various properties thereof.

Because of reasons of timing, no follow-up discussions occurred on this subject.

7. Recommendations for moving forward

Overall, there was a shared sense in the group that the proposed concepts can play an important role in the – to be devised – cross-repository interoperability layer. It is clear that the specifics remain subject to further discussion and experimentation, yet it was felt that a proper differentiation of each of these components has been reached. A brief summary is provided below:

- Several new concepts and terms have been proposed. A list of terminology is provided in Annex. It has been argued that research needs to be done on the reconciliation and interplay between these concepts and the concepts that are currently used in the World Wide Web Architecture.
- There was a reasonable amount of consensus on the attractiveness of the concepts of a surrogate. Though, several questions were raised on 1) the definition of core properties of a surrogate, 2) the inclusion of datastreams in a surrogate, 3) the difference between properties of a surrogate versus properties of the underlying content, and 4) the Intellectual Property associated with surrogates.
- There is a shared agreement in the group on the need for harvest and obtain repository interfaces. Though, some questions were raised about their actual implementation. For example: should we use OAI-PMH and NISO’s OpenURL; or should we create a new technology centered on ListIdentifiers and obtain.
- There was extensive discussion about the necessity of a put interface. In general, it was felt that it would be useful to have interfaces or tools that assist in populating local institutional repositories. Little exploration has been done on the use of put interfaces for cross-repository scenarios. It was also clear that the right name for this kind of functionality is not ‘put’; but rather ‘queue for ingest’ or ‘deposit request’.
- During the meeting, it was argued that the interoperability framework may benefit from the introduction of a additional services, including search and publish-subscribe. In general, it was felt that – while both sets of interface are desirable – they are not necessarily be part of the core ‘plumbing’ of each digital repository.

- There was little discussion about infrastructure and applications, although there is a shared sense in the group that some registries will be needed, possibly including registries of services, formats and semantics.

The participants at the meeting feel that this work is of vital importance and dedicated attention must be allocated to this work in order to ensure steady, focused progress. Along these lines, a possible path forward would include the following actions:

- A Steering Group would build on these discussion by writing a set of straw man specifications on the data model, the surrogate and a set of core services. The specifications should be lightweight and simple, yet solid, and conformant reference software should be developed.
- A Steering Group would coordinate a set of experiments that evaluate the straw man specifications and provide feedback to a meeting similar to the one conveyed here. Communities that we might consider engaging are the chemistry research community, the (bio)medical research community [with PubMed Central, National Library of Medicin's Entrez and Wellcome Trust], the Astronomy community [with the National Virtual Observatory], the meteorological and oceanographic community [with the NERC data grid, EcoGrid and NOAA], the cultural heritage, art museum, and archaeological communities, social history applications, and so forth.
- A Steering Group would further explore and work up some of the ideas that have been explored during the meeting and lay them out in a more systematic way, recognizing the discussion. In particular the concepts of queue-for-ingest and subscribe need to receive serious attention.
- *"We should engage the user community, so we do not create technical standards, for the sake of the technical exercise." (Tony Hey)*

Tony Hey also pointed out the similarities of this work with the effort of creating the Message Passing Interface (MPI). *"There, a process was started to create a few technical specifications. Lots of disagreement in the beginning, but at the conclusion of the process (and after having received lots of feedback), we reached a number of compromises. It is not the most elegant standard, but it works: There is an interoperable specification that people can pick up and run."*

The work that is object of this meeting requires a much more complex process. We haven't done all the experimentation, yet. And there is always the danger of premature standardization. Hence, we need to explore the concepts presented at the meeting by means of several experiments. The experiments need to be sufficiently complex: They have to involve different types of repositories and different communities." (Tony Hey)

"Follow-up conversations among the sponsors, organizers and presenters of this meeting will take place. We will come up with a pathway that will further address these ideas in the coming months." (Don Waters)

8. References

Bekaert, J., Liu, X., Van de Sompel, H., Lagoze, C., Payette, S., & Warner, S. (in press). Pathways Core. A Content Model for Cross-*Repository* Services. Submitted to the Joint Conference on digital Libraries, JCDL 2006.

Kahn, R., & Wilensky, R. (1995, May 13). A framework for distributed *digital object* services.

9. Annex (Terminology)

digital object: A digital object is a data structure whose principal components are digital data and key-metadata. Digital data can be a datastream or a digital object, i.e. a digital object may have one or more other digital objects as nested components. Key-metadata must include an identifier for the digital object.

datastream: A datastream is an ordered sequence of bytes.

data model: A data model is an abstraction for digital objects such that each digital object can be seen as an instance of the class defined by a data model. Example data models include the Pathways Core model, the MPEG-21 Digital Item Declaration model, etc.

surrogate: A surrogate is a serialization of a digital object according to a data model. The motivation for having a surrogate is the need for a common way of expressing digital objects at the repository interfaces (obtain, harvest, put) such that they can be accessed and leveraged by cross-repository services.

repository: A repository is a networked system that provides services pertaining to a collection of digital objects.

obtain interface: An obtain interface is a repository interface that supports the request of services pertaining to individual digital objects (including their component datastreams).

harvest interface: A harvest interface is a repository interface that exposes surrogates for incremental collecting/harvesting.

put interface: A put interface is a repository 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.