Update to the NDIIPP Architecture

Version 0.2 -- Draft for Outside Comment

Contents:

1. Introduction

This document outlines the current state of thinking on the Technical Architecture for the National Digital Information Infrastructure and Preservation Program (NDIIPP), following a period of review from April to July 2003.

The document outlines the updates and improvements to the architecture suggested by reviewers from various stakeholder communities and provides an updated high-level diagram of the interrelation of functions within the system.

Though the document in general reflects a refinement of the original goals and design, there is one especially noteworthy expansion of the goals. The meetings made vividly clear the remarkable energy going into work on digital preservation among a variety of institutions. As a result, the architecture has been reconsidered to take into account the need for interfaces between institutions to export and import not only digital objects but also whole collections and for institutions to be able to perform different roles in the system at the same time. (This change is noted in Section 6.1 below.)

We believe that this document will serve as the basis for conversations between the Library and other preserving institutions as we move into the execution phase of NDIIPP, and we do not doubt that there will be further articulation of architecture as a result of real-world experience gained during these efforts.

2. Background

In April and early May 2003, the Library of Congress and Global Business Network convened two meetings, one in Berkeley, Calif., and one in New York City, to solicit detailed feedback on the architecture for the National Digital Information Infrastructure and Preservation Program (NDIIPP). That meeting was followed by a revised document, which was further critiqued by a smaller group of practitioners on May 30 and again on July 28 and 29, in New York City. This synthesizes the feedback from those three meetings on the original architecture (hereinafter referred to as the 0.1 architecture) and served to produce this updated version (0.2) to circulate for further comment.

In addition to updating the diagram itself, this document also attempts to update the goals of the architectural work. While partnership with other institutions has always been part of NDIIPP, the April meetings made two things clear: The number of systems for digital preservation in existence and ready to test is growing rapidly; and there is strong desire for federating preservation across those systems or otherwise creating ways for those systems to interoperate.

Because of the variety of efforts on digital preservation, however, it is equally clear that there will be no quick convergence of methods in the digital preservation community. Every system is rightly designed to fulfill the goals of the sponsoring institution, and as institutional goals differ, so do the systems. While this does not damage the goal of digital preservation (in fact, it enhances it, because heterogeneity guards against systemwide failure), it also means that the trivial interoperability of “everyone uses the same tools and formats” and the deeper interoperability of “everyone uses the same conceptual model” are both unattainable, now and for the foreseeable future.

Because this sort of simple interoperability is outside our grasp, the NDIIPP architecture must support institutions that are inclined to cooperate with one another on issues of digital preservation, but who have differing technological systems in place.

3. Core Characteristics

A key attribute of the NDIIPP architecture is that it provide a bridge between disparate conceptual domains encapsulated by various existing and future systems. To do this, we believe that it must have two characteristics:

> First, it must describe the minimal set of functions required for digital preservation in such a way that existing systems can be mapped onto the architecture and vice-versa. A survey of the existing literature makes it clear that, although there is a common subset of functions required for preserving digital materials, the arrangement and even the names of those functions differ from system to system. The architectural model of the NDIIPP is not intended as a complete alternative version of existing

> systems. Instead, it is a kind “minimum requirement” set, designed to allow the Library to evaluate and compare real-world solutions.
>
> Second, it must not overspecify. Any complete system for digital preservation will have functions specific to the content or format of the material it is preserving (e.g., scholarly journals, digital films), as well as processes for supporting community-specific goals (e.g., evidentiary provenance, scholarly annotation). A system that attempted to be a superset of all such functions would be hopelessly bloated. A system that adopted one complete specification to the exclusion of others would not be sufficiently general to account for all the cases in which the Library has an interest.

This document attempts to describe the common functions and relationships necessary to describe the systems of preserving institutions that want to work cooperatively. It can be thought of as a kind of contractual rider, setting out a minimum understanding between two parties who are going to share the effort of digital preservation.

4. Terms

There are a handful of terms used in this document that serve as technological primitives on which higher-order definitions will be built. These terms are briefly noted here and defined in some detail in Appendix A of this document.

Identifier – A label for an object within the system. It does not necessarily specify a location of content within the system. An ISBN is a type of identifier.

Pointer – A reference to an identifier. A URL is a type of pointer.

Object – Anything stored in the system that has a pointer.

Unit – The smallest kind of object contained in the system. A unit is an object that contains no other objects (analogous to a file in a file system).

Container – An object that contains other objects, whether units or other containers or both (analogous to a folder in a file system).

5. Naming of Layers

We have renamed the layers in the 0.2 architecture. The 0.1 layers were called Interface, Collection, Gateway and Repository. The Gateway functions of security and metadata management have been redistributed throughout the system. As a result, the Gateway is no longer required as a separate layer.

The other layers have been renamed, on the grounds that the use of descriptive words (in archive, collection, repository, etc.) make conversation among organizations more difficult because those words often have specific and incompatible meanings for different organizations. To minimize this problem, the layers described in this document are now called Upper, Middle and Lower. Upper is the layer closest to the end user, however defined; Lower is closest to the storage of the digital objects; and Middle is where most preservation services exist.

6. Key Updates to the NDIIPP Architecture

We presented the 0.1 version of the architecture (outlined in Appendix 9 of the report to Congress and available at www.digitalpreservation.gov) to representatives from the technology, academic, archival and library communities. The meetings produced a wealth of valuable critique and useful recommendations. While much of the feedback was in the direction of identifying practical experiments, a significant portion of the feedback resulted in updates to the architectural model itself.

The most important of those updates are listed here:

1. The Possibility of Near-term Federation

   The meetings made very clear the remarkable energy going into work on digital preservation among a variety of institutions. Though working with other organizations was always part of the NDIIPP mandate, these meetings made it apparent that lightweight federation of existing efforts would be a valuable area of near-term exploration.

   As a result, the architecture has been reconsidered to take into account the need for interfaces between institutions to export and import not only digital objects but also whole collections and for institutions to be able to perform different roles in the system at the same time, as in the case of an organization that both holds a collection of its own (acting as at least a Middle layer) and serves as a repository for other collections (acting as a Lower layer for other institutions.)

2. Metadata Exists at the Lower (Repository) Layer

   The 0.1 architecture imagined a relatively clean separation of data and metadata (other than location) at the Repository layer (now renamed the Lower layer) accomplished by a redirection function at the Gateway layer.

   The two review meetings, and in particular the New York meeting, made the unworkability of this notion clear. The consensus view was that at least some metadata will be mingled with the objects, for two reasons: first, several file formats contain internal specifications in their own headers (e.g., JPEG), making inclusion of at least some metadata automatic. Second, the strong opinion of reviewers was that storing only raw data created unacceptable risk should the data become even briefly unmoored from a preserving institution.

However, the alternate extreme – all data should be stored with all of its metadata – is also unworkable, lest the works of Plato, for example, have to travel with all written commentary since. We assume that every preserving system will find its own methods of commingling data and metadata, and that the principles for such commingling will be specific to the institution, the preservation system, collection or collections and the data to be preserved.

We thus make no requirements about the type, amount or method of storing data and metadata together; we expect later work on standards and practices to suggest that storing at least the technical metadata required for interpretation or playback should be standard practice.

3. Metadata Evolves Over Time

We likewise assume that over the course of the life of an object, additional metadata in the form of annotations and descriptions of administrative actions will accrue in the Middle layer. And we assume that each individual organization will have to make judgments about when to attach this metadata to the object itself, when to make a new edition or other copy of the object replete with new metadata and when the metadata itself needs to be archived as a digital object (as with the contents of a card catalog, for example.) We therefore specify in the system only that a stored object will be accompanied by at least some metadata and that it is unlikely that any object will ever be accompanied by all possible metadata. We do not describe when or how additional metadata should be generated or stored.

4. A Container Is Also an Object

Digital objects are often internally complex – word processing documents and Web pages can contain internal graphics, but these graphics are embedded in different ways and may be directly addressable on their own. There can be an unlimited number of such containers – a Web site is a collection of complex files, the results of a Web crawl may be several Web sites, and so on.

The only lower limit to the granularity of objects is the end of containment. An object in the system that contains no other objects is called a unit. All higher-order objects (objects that contain at least one other object) are containers, and, like units, containers have unique identifiers within the system.

5. The Diffusion of the Gateway Layer Functions

   The 0.1 architecture included a Gateway layer, whose role was to separate data and metadata and to provide access control to storage. The functions of the Gateway layer have been redistributed, for two reasons. First, as discussed above, data and metadata cannot not be completely decoupled. Second, the weaker the security of a system, the greater the risk of outside interference; but the stronger the security, the greater the cost.

   It became clear from feedback on the 0.1 architecture that any particular specification of security practices would be too weak for some users but too expensive for others. Given that there is no need for a Gateway layer for metadata management, the 0.2 architecture assumes that security will be suffused throughout the system, whether as encrypted files, on-the-wire encryption, access control or other methods, singly or in combination, on a case-by-case basis.

6. Functions at the Middle (formerly Collection) Layer

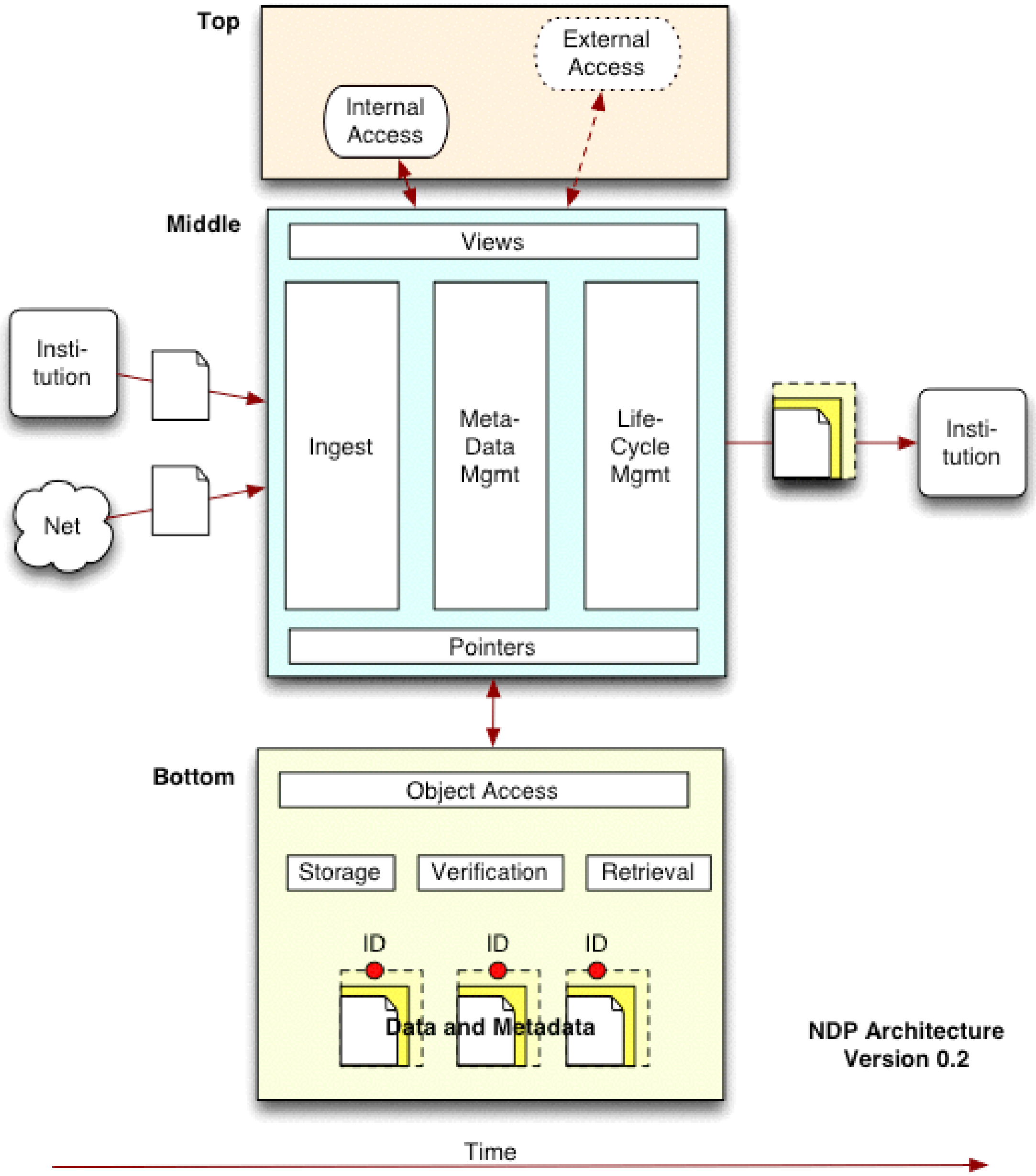
   The Collection layer in the 0.1 architecture contained too many functions in an opaque container to serve as a useful abstraction. This version of the architecture, 0.2, breaks the Collection layer, now renamed the Middle layer, into five critical categories of functions: Ingest, Pointer Management, Metadata Management, Lifecycle Management and Views. These functions are defined below, in the section on the new architectural diagram.

7. “Local” vs. Public Access at the Upper (formerly Interface) Layer

   The 0.1 architecture treated access as a binary condition – public access or dark archive -- with the notion that some public access might be limited by terms and conditions set by collecting agencies. Feedback from respondents suggested that, in addition to addressing questions of public access, every set of digital objects needs to provide local access to the management responsible for the preservation of those objects. The Upper Layer now reflects both kinds of access.

7. Architectural Diagram, Version 0.2

Included below is the updated diagram of the NDIIPP architecture.



At the left, digital material passes into a preserving institution, whether the material was donated by a person or institution or automatically accessed as with a Web crawl.

In the center are the functions of the preserving institution or institutions. On the right is the export of material from the preserving institution outward. Note that this export can be in frequent small batches or in periodic snapshot exports of an entire collection. Note also that the architecture assumes that data is exported in a format that packages the object with some additional metadata.

Down the center of the diagram are the functions of a preserving institution.

8. Descriptions of Layers

1. Lower

At the Lower layer are the services required for storage, verification and retrieval of digital objects, as defined above, whether for containers or units. A unit is assumed to be a digital object accompanied by at least some of its metadata. It is also assumed that the "halo" of metadata around an object will grow over time (indicated by the dotted-line container) as additional interpretive or provenance data is registered.

The Lower layer is a group of functions, but is not necessarily an integral piece of technology. The object access interface presents a coherent view of the stored objects, but beneath that interface data can be stored in a number of distributed or virtualized ways. A database, for example, may well be able to present a combined view of data and metadata, but it may store the various elements in separate tables on separate disks. Likewise, the de-referencing of pointers to IDs that specify physical location of digital material may go through several layers of redirection if the files are stored in multiple copies or chunks on decentralized and geographically dispersed systems (e.g., LOCKSS or OceanStore).

2. Middle

The Middle layer contains five functional categories:

1. **Ingest** – the functions required for the transfer of responsibility for the preservation of digital data to a particular organization, including both the acceptance of digital materials and the creation of any contractual or administrative agreements.
2. **Pointer Management** – the creation or registration of pointers for the digital objects being preserved. Pointers point to digital objects stored in the Lower layer.
3. **Metadata Management** – the creation and management of metadata for the digital objects being preserved. (Note that at least some metadata will be stored with the object itself). At a minimum, this metadata will include or point to as much detail as possible on making the object available for interpretive use – file format, conditions of creation, playback software, etc. Note that the metadata can be stored by other institutions, including third-party service providers, as well as by the hosting institution.

   Note also that additional metadata will be developed over time, in forms ranging from additional management or scholarly annotation to administrative notes related to the management of the object.

4. **Life-cycle Management** – the set of operations required to make digital data fit for use over the passage of time, including the transfer of copies of the original objects in bit-identical format onto new storage media; the migration of objects to new formats; the documentation of emulation strategies for playing back older data on newer software; and the export of objects, which entails the possible transfer of metadata and of preservation responsibility, if it is contractually agreed upon, to other preserving institutions.

5. **Views** – The Views function essentially plays a gatekeeper role for the provision of access to the objects, filtered through whatever policies or restrictions are placed on their use (available internally only or available to other institutions), any particular file transformations that are allowed or disallowed, etc.

This is not to say that functions in the Upper or Lower layers cannot also be coupled with these functions in a single organization or even on a single machine, nor is it to say that additional functions cannot be deemed essential by individual organizations. These functions are grouped together because they are essential and relatively difficult to decouple – ingest requires both pointer and metadata management; meta-data must be associated with objects identified by pointers; life-cycle management operates on objects identified by pointers and generates new metadata; and so on.

The five categories in the Middle layer encapsulate a wide range of functions – any working system will have to break out those functions in more detail. However, a survey of the literature suggests that the next level of detail is where existing systems begin to diverge.

In the realm of metadata management, for example, the OAIS reference model refers to Preservation Description Information, comprising Provenance, Reference, Fixity and Context Information. METS, by contrast, categorizes metadata into three categories, descriptive, administrative and structural, with administrative metadata further subdivided into source, technical, intellectual property and provenance metadata. Thus, the five functions listed here are an attempt to outline minimal required functions while providing a description sufficiently general to apply across a range of practical implementations.

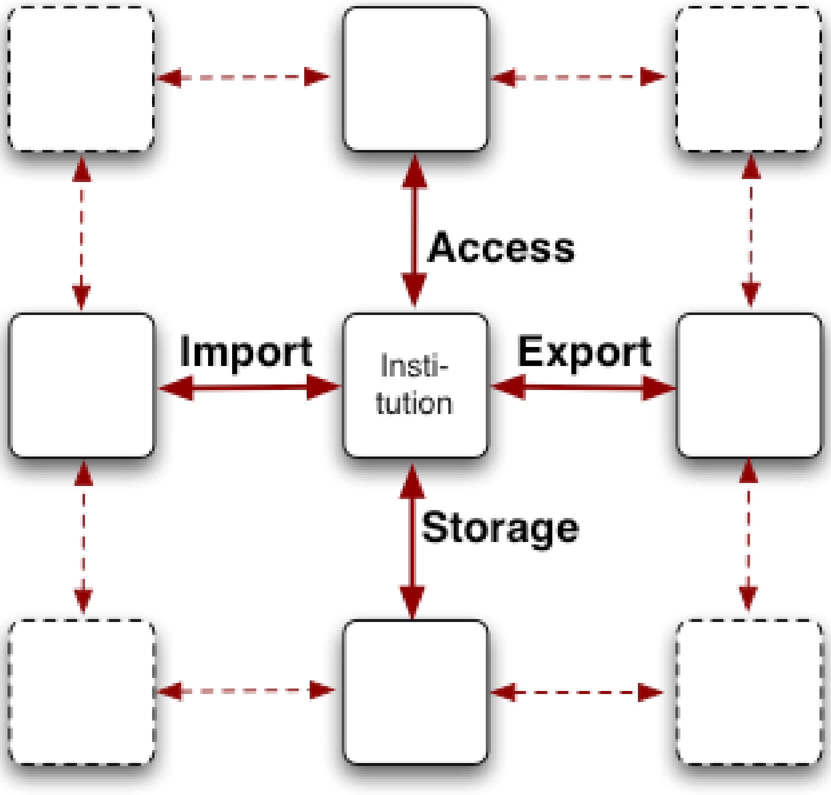
3. Upper

The Upper layer comprises access by any person or institution to data or metadata through the Views function of the Middle layer. There are two broad categories in the Upper layer – internal access and external access. Internal access describes any human views of the system required for management of the material, from creation of descriptive metadata to spot checks for validity or interpretability of the content.

As in the 0.1 architecture, the Upper layer is minimally and mostly negatively defined. Because the Middle layer is where the hard work of maintaining enough metadata to allow the object to be reconstituted lies, the principal requirement of the Upper layer, whether through internal or external access, is to provide some form of trustworthy mediation for potentially untrustworthy users, in the case of sensitive or restricted materials and, in any case, not to violate any legal or administrative controls set upon the data.

9. Relationship Between Interfaces and Layers

As with any system designed around nodes and connections, the 0.2 architecture can also be viewed in an interface-oriented fashion. The drawing below presents the system as a set of four interfaces arranged around the Middle layer – Import, Storage, Access and Export – representing the possible interfaces a preserving institution might have with the outside world.



These four interfaces all have related functions within the Middle layer: Import/Ingest, Storage/Pointers, Access/Views and Export/Life cycle. These interfaces are labeled from the point of view of the center institution, but the functions are arranged in parallel: one institution's Export function connects to another’s Import, and one institution’s Access function connects to another’s Storage.

This view, while related to the function-centric view above, highlights systemic aspects of NDIIPP.

- Interface definitions between institutions are a critical feature of a working system.
- The functions taken on by preserving institutions are complex but largely opaque. The complexity of managing a digital collection internally is (or should be) hidden from the outside world.
- Any institution can perform multiple functions within the system. In a scenario whereby institution A accesses content held by institution B, B is a de facto Lower layer for A, even if B also offers direct access to the same material. There is no theoretical upper limit to this sort of redirection, though practicality suggests that most transactions will involve three or fewer institutions.
- Likewise, an institution can import content from other institutions and export it to other institutions.
- Vertical interfaces – access and storage –work relatively well in current systems and have analogs in everything from networked file systems to the Web and Web Services architectures.
- Horizontal interfaces currently work less well. Ingest in current systems tends to be human-intensive and therefore expensive for bulk accessioning of data. Likewise, the ability to export a complex collection of digital objects in an archival format is limited, and an important area of future work.

The most important design principle of the interface view of the system is that, although the functions are divided into three layers, the system as a whole is an N-layer system, because it is impossible to specify in advance how participating institutions will stand in relation to one another over time.

10. Conclusion

Though the proposed 0.2 version of the NDIIPP architecture preserves the basic design principles from the work on 0.1, especially the goal of a modular and protocol-connected architecture, the 0.2 version is simultaneously simpler and more detailed: simpler because the decision not to overspecify metadata management and the subsequent removal of the Gateway layer make the conceptual units of the system easier to understand and to map to existing efforts; more detailed in that the functions within the layers, and especially the Middle layer, are better specified.

The acid test, of course, is whether some version of this document will be useful in brokering conversations between the Library of Congress and other institutions engaged in preservation activities, as well as among those institutions themselves. To that end, we are actively seeking feedback on whether this document captures the minimal set of functions required for preservation activity. In particular, we are looking for feedback on missing but required functions, included but nonessential functions and places where the level of detail can be made more specific without entailing the loss of generality required to describe the intersection of most or all existing preservation systems.

Appendix A: Terms

There are a few terms used in this document that serve as technological primitives, on which higher-order definitions will be built.

Identifier – A globally unique and persistent label for an object within the system. It does not necessarily specify a location of content within the system. An ISBN is a type of identifier.

Pointer – A reference to an identifier. A URL is a type of pointer. Note that sometimes Identifiers and Pointers can be identical, as when a URI that is also a URL, while in other cases they can be separate; isbn.nu uses ISBNs in its URLs, but an ISBN is different from an isbn.nu URL. The critical point is that a digital object must not just be labeled; there must be some way to refer to that object remotely, through one or more pointers.

Every functional system must provide a method of de-referencing pointers to the identifiers (and thus to the objects) it is responsible for preserving, even if that de-referencing goes through layers of redirection.

Version control can be, but is not required to be, part of the identifier/pointer system. An identifier or a pointer can include explicit methods of version control by providing operations such as incrementing or decrementing counters or alternating MIME types to access other versions of the "same" object, or the pointers to earlier versions can be stored elsewhere in the metadata of the object referenced by the pointer.

The current architecture is mute on this question not because it is unimportant, but because we do not believe that there is general agreement on the versioning issue among extant systems. Part of the next phase of NDIIPP work will be to experiment with strategies for version control.

Object – An entity in the system with a pointer. The two types of objects are units and containers. These are roughly analogous to files and directories, with the key difference being that there is no required “root” object.

Unit – The smallest object contained in the system. A unit is an object that contains no other objects, though it may contain pointers to other objects, e.g., a Web page that has pointers to images.

Container – An object that contains other objects, whether units or other containers or both. A container can be as tightly coordinated as a file that includes embedded images and as large as a container-of-containers that encompasses the entire holdings of an institution. There is no mandated “root” container and no upper limit to the number of layers of containment possible.

The goal is to know, in principle, what is contained in a container. When containers are imported or exported, they may travel with their contained contents (including, of course, other containers) or they may simply be passed as metadata (and updated pointers, if needed), with the bulk of the contained objects being left in place.

Note that managing Web pages is particularly problematic, as Web pages embed other first-order objects with URLs (as with images or the content of frames), rather than direct inclusion, as with a PDF. Web pages are therefore unbounded as objects, even in principle, as they can always point to new resources for inclusion. Further work is needed to know whether to treat Web pages as units, containers or as a special third category.