A Data Model and Architecture for Long-term Preservation

Greg Janée
Map & Imagery Laboratory
University of California,
Santa Barbara
+1 (805) 893-8453
gjanee@alexandria.ucsb.edu

Justin Mathena
Map & Imagery Laboratory
University of California,
Santa Barbara
+1 (805) 893-5452
mathena@library.ucsb.edu

James Frew
Donald Bren School of Environmental
Science and Management
University of California, Santa Barbara
+1 (805) 893-7356
frew@bren.ucsb.edu

ABSTRACT
The National Geospatial Digital Archive, one of eight initial projects funded under the Library of Congress's NDIIPP program, has been researching how geospatial data can be preserved on a national scale and be made available to future generations. In this paper we describe an archive architecture that provides a minimal approach to the long-term preservation of digital objects based on co-archiving of object semantics, uniform representation of objects and semantics, explicit storage of all objects and semantics as files, and abstraction of the underlying storage system. This architecture ensures that digital objects can be easily migrated from archive to archive over time and that the objects can, in principle, be made usable again at any point in the future; its primary benefit is that it serves as a fallback strategy against, and as a foundation for, more sophisticated (and costly) preservation strategies. We describe an implementation of this architecture in a protoype archive running at UCSB that also incorporates a suite of ingest and access components.

Categories and Subject Descriptors
H.3.7 [**Digital Libraries**]: Systems issues; E.2 [**Data Storage Representations**]: Object representation.

General Terms
Design, Standardization.

Keywords
long-term preservation; curation

1. INTRODUCTION
The accelerating increase in the amount of digital information, coupled with the aging of our existing digital heritage and well-publicized examples of its loss, have brought a sense of urgency to the problem of long-term preservation of digital information. Recent years have witnessed the development of preservation-supportive repository systems, format registries, and tools.

JCDL'08, June 16–20, 2008, Pittsburgh, PA, USA.

It is already clear from this initial research and development that there will be no simple, "silver bullet" solution to the problem of preservation. Preserving our digital heritage will require a complex and evolving mixture of investment in curation and upkeep, careful standardization, development of new kinds of information systems, and automation of processes such as provenance tracking and format migration.

Each type of information, whether it be text or music or imagery, brings to the preservation problem its own complications and special requirements. Geospatial data—the wide variety of scientific and government-produced datasets that have a geographic component—poses unique challenges to preservation because of its size and complexity:

- Geospatial information is voluminous. Like some multimedia datasets (e.g., movies), geospatial datasets may have gigabyte granularities, so that preservation and migration decisions at the level of single objects can measurably impact system capacities (storage, bandwidth, etc.).
- Geospatial datasets often have a time dimension. The largest geospatial datasets, generated by Earth satellite sensor systems, grow at rates of up to terabytes per day, often for years. This has two challenging consequences. First, some geospatial datasets can grow so rapidly that it becomes necessary to begin archiving them before they are "finished," i.e., while new information is still being added to the dataset. Second, some geospatial datasets may continue growing for longer than an archive's optimum technology refresh period. Traditionally this has been addressed by binding large datasets to obsolete storage systems, an unsupportable strategy in the long-term.
- Geospatial information is highly structured. Additional structure is often imposed on geospatial data in order to compactly represent its dimensionality (e.g., by establishing a correspondence between space-time coordinates and addresses in a multidimensional array) or attribute domains (e.g., by encoding measurement scales or categorization schemes). These structures may be at least as complicated as common text formats (e.g., PDF), but are far less ubiquitous, and may have only proprietary realizations. Archival policies for geospatial information may require the preservation of tools and/or procedures as well as format descriptions.
- Geospatial information needs special interpretation. Interpreting geospatial information often requires special knowledge not widely embedded in the source culture. For example, ensuring the survival of a text

document may be satisfied by ensuring the survival of a capability to sensibly render it—the archive may safely assume that the ability of (some) humans to read the document will survive independently. By contrast, geospatial information often requires much more than simple rendering (e.g., of a satellite data granule as an image) for its interpretation. Highly specialized and detailed characterizations of the information's sources, metrics, and processing history must be preserved, making geospatial archival objects necessarily more complicated than traditional archival materials.

- Geospatial information is tightly bound to specific models of the real world. These models include both explicit (geocentric or projected coordinate) and implicit (place name or description) locations, often compounded with a time reference. These models must be accurately characterized and preserved along with the information.

Despite these complications, the need and desire to preserve historical geospatial data is great given the extensive and vital role it plays in scientific, government, and legal communities.

The National Geospatial Digital Archive (NGDA) [15] is one of eight initial projects funded by the Library of Congress's National Digital Information Infrastructure and Preservation Program (NDIIPP). NGDA participants include the Map & Imagery Laboratory at the University of California at Santa Barbara and Branner Earth Sciences Library at Stanford University. The project's overarching goal is to answer the question: How can we preserve geospatial data on a national scale and make it available to future generations?

In this paper we describe our architecture for federated archives of geospatial data. As will be seen, this architecture is simple and conservative, and is designed to provide only a fallback strategy that guards against irreplaceable loss of information. However, the architecture is generic enough to serve as a foundation for more sophisticated (and costly) preservation strategies, for both geospatial and other types of information.

2. PRESERVATION PRINCIPLES

Four observations about long-term preservation influenced the development of the NGDA architecture.

2.1 Preservation is a relay

In the context of preservation, we define "long-term" to be a period of time exceeding the lifetimes of the people, applications, and platforms that originally created the information. For the sake of argument, we will use 100 years as a reference timeframe. Can we design an archive system that will preserve information for 100 years?

The answer is most assuredly negative; no system can reasonably be expected to last 100 years. In our experience in building and running digital library and storage systems, a ten-year-old system is nearing the end of its lifetime in terms of supportability. (The US Internal Revenue Service operates archive systems that are multiple decades old, but this is newsworthy precisely because it is so unsupportable and beyond the norm [10].) Storage systems change much more frequently, with turnovers in technology occurring every 3–5 years. Even entire institutions come and go. Few institutions can guarantee their continued existence, let alone support for preserving a specific piece of information, for 100 years; and that still leaves the issue of changes in curatorship within institutions.

As a consequence of this general impermanence, long-term preservation is best characterized as an extended *relay* in time, with information being handed off from storage system to storage system, from repository to repository, and from institution to institution [6]. Thus a key consideration in the design of an archive system is not just how well preservation is supported over the archive's lifetime, but how well and how easily the archive can hand off its contents and responsibilities to the next archive in an ongoing succession.

Furthermore, future archives must be prepared to work with old digital information. Given our short digital history, most archives today are in the fortunate position of working with recently created information; that is, with information types that are still current and well-understood in their respective communities. But if we consider our 100-year reference timespan, archives in the middle of that span will be faced with curating information for which all links to the original creators and context have been severed. (Consider, in 2008, curating digital materials created in 1958.) Thus we can strengthen our previous conclusion: a key consideration in the design of an archive system is not just that it can curate information and then hand the information off to the next archive, but that it can do so for unfamiliar information types, and in such a way that the next archive can make the same claim, and so on through time.

2.2 Preservation is mitigation of risk

It is common to speak of so-called "persistent" identifiers and "persistent" objects, but strictly speaking, persistence is not an attribute of a piece of information; rather, persistence is an *outcome*, a result of the past commitments of the information's curators and holding institutions [14]. At best, future persistence hinges on our ability to mitigate known risks to persistence.

There are many such risks to be addressed, from storage loss to format obsolescence, but an overriding risk is lack of resources to properly curate information. For a given piece of information, we cannot assume that the information will be maintained in a usable state at every point in its existence: the perceived value of information changes over time; archive resources inevitably change over time; and there may be points in time at which the upkeep of the information cannot be supported or justified. The assumption that sufficient curation resources will always be available over a piece of information's lifetime is a risk that must be mitigated.

The risk of insufficient resources is particularly acute when handoffs occur (as described in section 2.1), particularly handoffs between archive systems and between institutions. These handoffs may occur precisely because the previous archive system becomes unsupportable, or the current institution runs out of resources, resulting in a situation that puts immediate strain on the receiving institution.

As a consequence, a key consideration in the design of an archive system is that there be an ultra-low-cost, *fallback* preservation mode that, while perhaps not maintaining the information in any immediately usable state, nevertheless preserves it well enough that the information can, in principle, be "resurrected" at any time in the future given sufficient desire and resources. This has been colloquially referred to as an archive's "option to do nothing."[1]

[1] Thanks to Clay Shirky of New York University's Interactive Telecommunications Program for this phrase.

2.3 Preservation of context is necessary

Long-term preservation requires solving two complementary, yet coordinated problems: preservation of the information itself (the raw "bits"), and preservation of the context surrounding the information (the meta-information needed to interpret the bits).

That the bits require preservation goes without saying. But context requires preservation, too, for two reasons. First, context is often hidden or implicit. For example, a webpage is renderable by any contemporary personal computer, not by virtue of any knowledge on the user's part, but by the fact that knowledge of the relevant file formats is embedded in the computer's operating system and web browser application. Second, the context in which information is produced is constantly evolving. The *technological* context surrounding any piece of information—the computing platforms, programming languages, applications, file formats, and so forth—will inevitably change over time until the information is no longer usable. Changes in the information's *social* context are just as significant. The communities and organizations involved in the information's creation and initial use may attach different values and interpretations to the information over time, or cease to exist altogether. The combination of context being both implicit and evolving requires that, for information to remain usable, either the information must continuously evolve along with the context, or the context must be captured and preserved along with the information, preferably at the time of the information's creation.

The context surrounding geospatial information is particularly complex, as enumerated in the introduction. For example, use of remote-sensing imagery requires detailed knowledge of sensor and platform characteristics, which, due to its size and complexity, is not usually bundled with data objects in the same way that descriptive metadata is. Furthermore, geospatial data may require deep analysis to remain usable over time. For example, to support long-term, longitudinal climate studies, historical climate data records must be periodically reprocessed to conform to the most recent revisions of scientific understanding and modeling. This in turn requires access to and understanding of the original processing, including scientific papers, algorithm documentation, processing source code, calibration tables and databases, and ancillary datasets [12].

We also note that whereas preservation of bits requires that the bits stay unchanged over time, preservation of context must accommodate and even embrace change to the context. File formats will need to be migrated over time, new access services will need to be developed and will require new kinds of support, and information will inevitably be reorganized and recontextualized.

Thus a key consideration in the design of an archive system is that it be able to capture and preserve complex contextual information; maintain persistent associations between information objects and contextual objects; and support modification of the context over time.

2.4 Preservation must be as cheap as possible

An old engineering adage states "good, fast, cheap: pick any two." For long-term preservation the adage might be restated to say: "scale, longevity, economy: pick any two." We can preserve information on a broad scale; we can preserve information for long periods of time; we can preserve information cheaply; but we can't do all three. Given NGDA's goal to preserve geospatial data on a national scale for 100 years, that leaves economy as the unachievable goal. And yet, we must have economy, for there are few funding structures in place to pay for curation.

Another way to look at cost is to ask: Is a given information object worth preserving? To answer this question, a curator must in effect solve a complex equation that balances the perceived intrinsic value of the information and the probability of its usage in the future on the one hand, against preservation and storage costs and, in the case of fallback preservation, costs to resurrect the information and make it usable again, on the other hand. Significantly, the only variables in this equation under the control of the curator are costs, which can only be minimized.

As a consequence, a key consideration in the design of an archive system is its ability to support multiple levels of preservation and usability, the choice of which can vary in response to changing information values, preservation costs, and resource availability.

2.5 Requirements

Summarizing the preceding discussion, we conclude that to support long-term preservation, an archive system architecture must:

- Explicitly capture and preserve the context necessary to interpret information objects in the future.
- Provide multiple levels of preservation and usability to allow archives to adapt to varying resource availability.
- Maintain a fallback representation that provides for the survivability of archived content even in the face of repository and institution failure.
- Facilitate handoff to the next archive system.

3. ARCHITECTURE

The NGDA architecture is an attempt to satisfy the preceding requirements. The architecture, shown in Figure 1, is based on a data model that defines a uniform representation of all information in the archive, paired with a storage API that abstracts the storage subsystem. The storage subsystem is assumed to provide reliable, long-term storage through redundancy and active monitoring. A suite of components built on top of the data model and storage API provide ingest and access functionality. We describe each of these facets of the architecture in turn.

3.1 Data model

3.1.1 Logical data model

3.1.1.1 Fundamentals

The fundamental entity in the NGDA data model (Figure 2) is the *archival object*, which:

- represents one independently reusable piece of information;
- contains all components necessary to use the object, including data, metadata, and both original and derived forms; and
- has links to related archival objects.

An *archive* is a collection of archival objects.

An archival object is identified by a universally unique identifier (the identifier scheme is unspecified by the data model) and may contain any number of *components*, of which there are two types: files and directories. All components have a name that uniquely identifies the component within the archival object or enclosing component and, optionally, a description stating the role the component plays within the object.

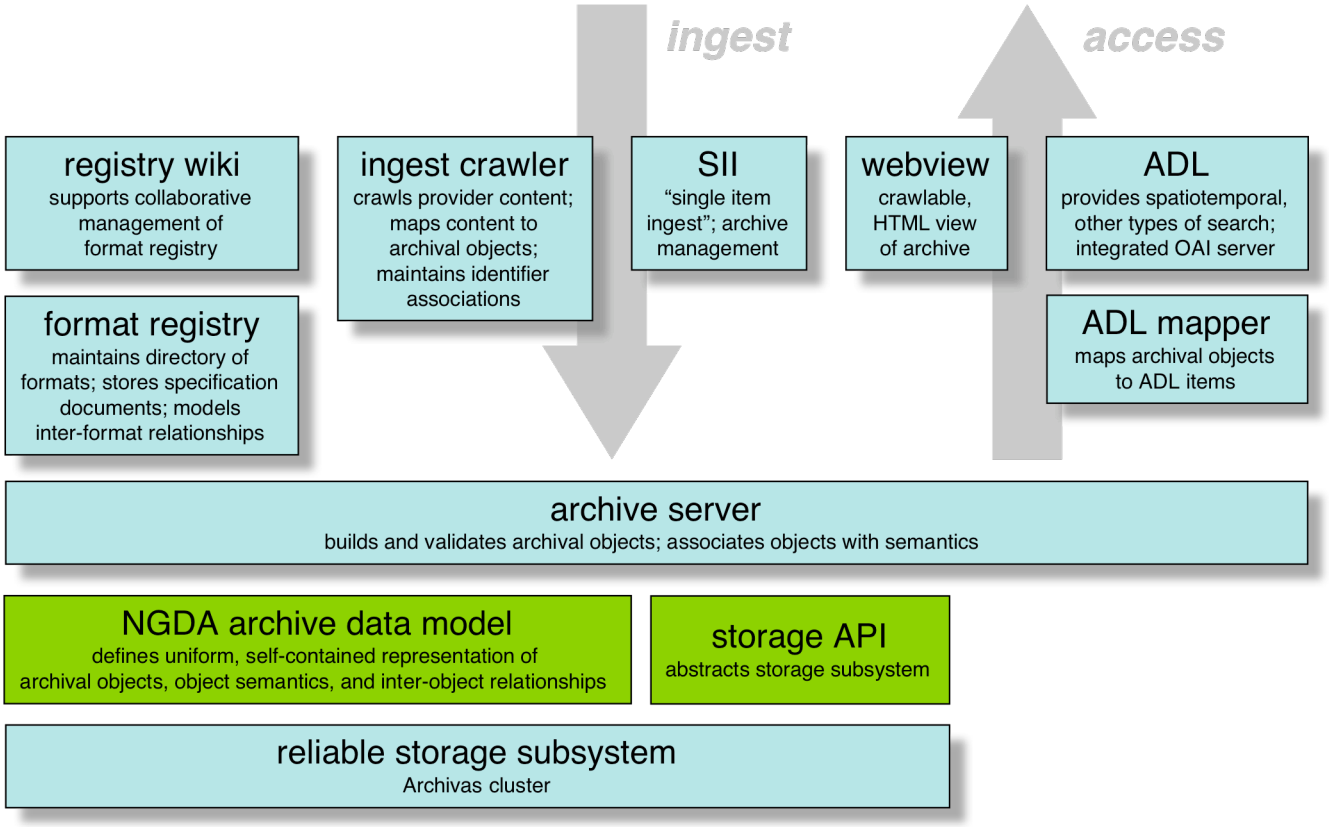

Figure 1. The NGDA architecture.

A *file component* represents a typed bitstream and holds the bitstream, the bitstream's size in bytes, and a content signature. The latter two items are, strictly speaking, not necessary because the NGDA architecture assumes that the storage subsystem is reliable, but they are explicitly included in the data model to facilitate reliable handoffs across archives and across storage systems.

Directory components are used to implement hierarchies, and come in two flavors: an "alternatives" directory describes different representations of substantially the same information (for example, JPEG and TIFF version of the same image), while a "subcomponents" directory describes an aggregation. In the authors' digital library work this distinction has been found to be useful [8].

An abridged example of an archival object representing a dataset of county boundary lines is shown in Figure 3. The object has two top-level components:

- an XML-encoded FGDC metadata record;
- two alternative representations of the object's data:
  - a Shapefile (a common GIS format which, despite its name, is actually a set of files); and
  - a PNG image.

We will continue to refer to this example in the following.

3.1.1.2 Definitions

Every component may have one or more *definitions* of its interpretation. A definition may be a file format, or any other type of semantic specification; structurally, a definition takes the form of a link to another archival object. Thus instead of describing file component formats by using, say, MIME types, the NGDA data model assumes format information is an integral part of the archive. The format of a file is indicated by a link to the archival object representing that format; the latter object holds the actual format specification documents along with format metadata recording MIME types, file suffixes, and so forth.

A link is represented by simply naming the target archival object's universally unique identifier. The data model does not mandate any particular identifier system. Thus, the particular identifier system employed by an archive should be considered to be one of its fundamental dependencies.

(Not reflected in the UML diagram in Figure 2 is the fact that an archival object as a whole may have one or more definitions.)

In the example in Figure 3, the FGDC metadata and PNG image files have links to their respective formats. The link to the Shapefile format is at the directory level because that format defines the interpretation of all Shapefile component files. A component without a definition is implicitly defined by the nearest ancestor definition.

3.1.1.3 Other features

A relationship is a named, directional association between two archival objects. There is no fixed vocabulary for relationship names, though standardization in this area would be welcome. Relationships may be used to record, for example, the fact that one object follows another in a series of like objects, or that one format is a subtype of another format.

Noticeably missing from the data model are additional kinds of entities for representing aggregate structures such as collections and series, or specialized object types for representing providers and formats. Instead, we represent these concepts by building on

the model elements described so far, namely, collections of components organized into archival objects. With this approach, an archival object representing a format or collection is recognized as such by virtue of containing appropriate metadata (format registry metadata or collection-level metadata, respectively) and by participating in appropriate relationships (such as format–format relationships in the first case, or collection–member relationships in the second). In the example in Figure 3, the archival object has a "member of" relationship to an archival object representing the CaSIL geopolitical collection. A reciprocal "has member" relationship may or may not be present.

The data model also incorporates a simple lineage model. Each component may have annotated links to the sources from which it was derived. The source may be another component within the same archival object, a component in another archival object, or simply another archival object. In Figure 3, the PNG image (described as a "preview") is noted as having been derived from the Shapefile. Not reflected in the UML diagram in Figure 2 is that fact that an archival object as a whole may have lineage links.

Finally, the NGDA project has developed some ingest-time aids that complement the data model so far described. An *ingest language* defines commands that can be used to build up archival objects from constituent pieces. A *template* is an incomplete archival object that can be referenced from the ingest language to simplify populating an archive with similar objects. For example, a template might contain a file component with a definition link to the FGDC metadata format, but omit the file's actual content and size and signature; the latter would be filled in at ingest time. These ingest-time aids are not integral to the data model, but have proven to be convenient.

3.1.1.4 Formal properties

The data model includes several formal properties that restrict the kinds of models that can exist. Two examples:

- Definition: Every component must be defined. A component is defined if it has at least one definition, or if an ancestor of the component within the containing archival object (or the archival object itself) is defined.
- Lineage: An archival object or component thereof must not be derived from a constituent component. Furthermore, the directed graph induced by lineage relationships must be acyclic.

Discussion. One of the main goals of this data model is to ensure that archived information is *interpretable*, that is, that the archive contains sufficient format and other contextual information so that use of an object can be reconstructed arbitrarily far into the future. The definition property raises the possibility of using the graph-theoretic nature of the data model to create a formal definition of interpretability which would facilitate automated validation. Let us define a component or archival object to be interpretable if its definition(s) are (recursively) interpretable. Thus we might be able to say that a JPEG image is interpretable because the JPEG format is interpretable, and the latter is interpretable because the JPEG specification, a PDF document, is interpretable because the PDF format is interpretable, and so forth.

However, we immediately run into problems with this line of formalism. Loops are possible: it turns out that the PDF format is defined by a PDF document. This particular case can be averted

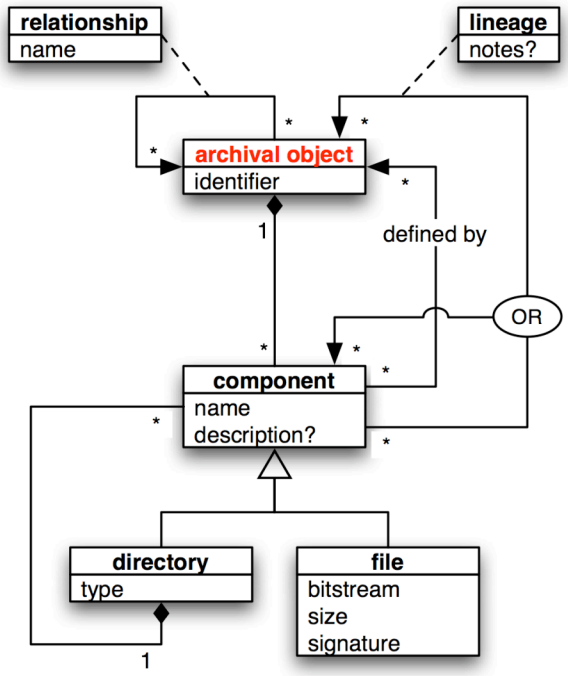

Figure 2. Simplified UML diagram of the NGDA data model.

by creating a "desiccated"[2] representation of the document, i.e., a representation that trades off usability for greater preservability. (As a demonstration, the NGDA format registry includes GIF format page scans of the PDF specification.) This leads to a second problem: the recursion must fail somewhere. The GIF format is defined by an ASCII text document, but how to define the ASCII character encoding?

Both the preceding problems could be averted by simply "blessing" certain "core" archival objects as being interpretable. But then numerous problems relating to relationships arise. The GeoTIFF format is defined by an HTML document, but as a subtype of TIFF, it can be understood only in the context of the latter specification. Hence, at minimum, our definition of interpretability must include the interpretability of parent types. But more ambiguous relationships also come into play. The RELAX NG format does not share any subtype relationship with XML, but as an XML schema language, it can be understood only in the context of XML. Hence, at least in some cases, related types must be considered. And then there is the problem that, in some but not all situations, interpretability of objects depends on metadata held in collection-level or series-level objects.

Given the difficulty of knowing which relationships to follow, we conclude that the concept of interpretability has a significant qualitative component that requires analysis by a curator.

3.1.2 Physical data model

Complementary to the logical data model is a physical data model that represents all information in the archive as files in a filesystem. In the physical model, an archival object is

[2] Thanks to John Kunze of the California Digital Library for this term.

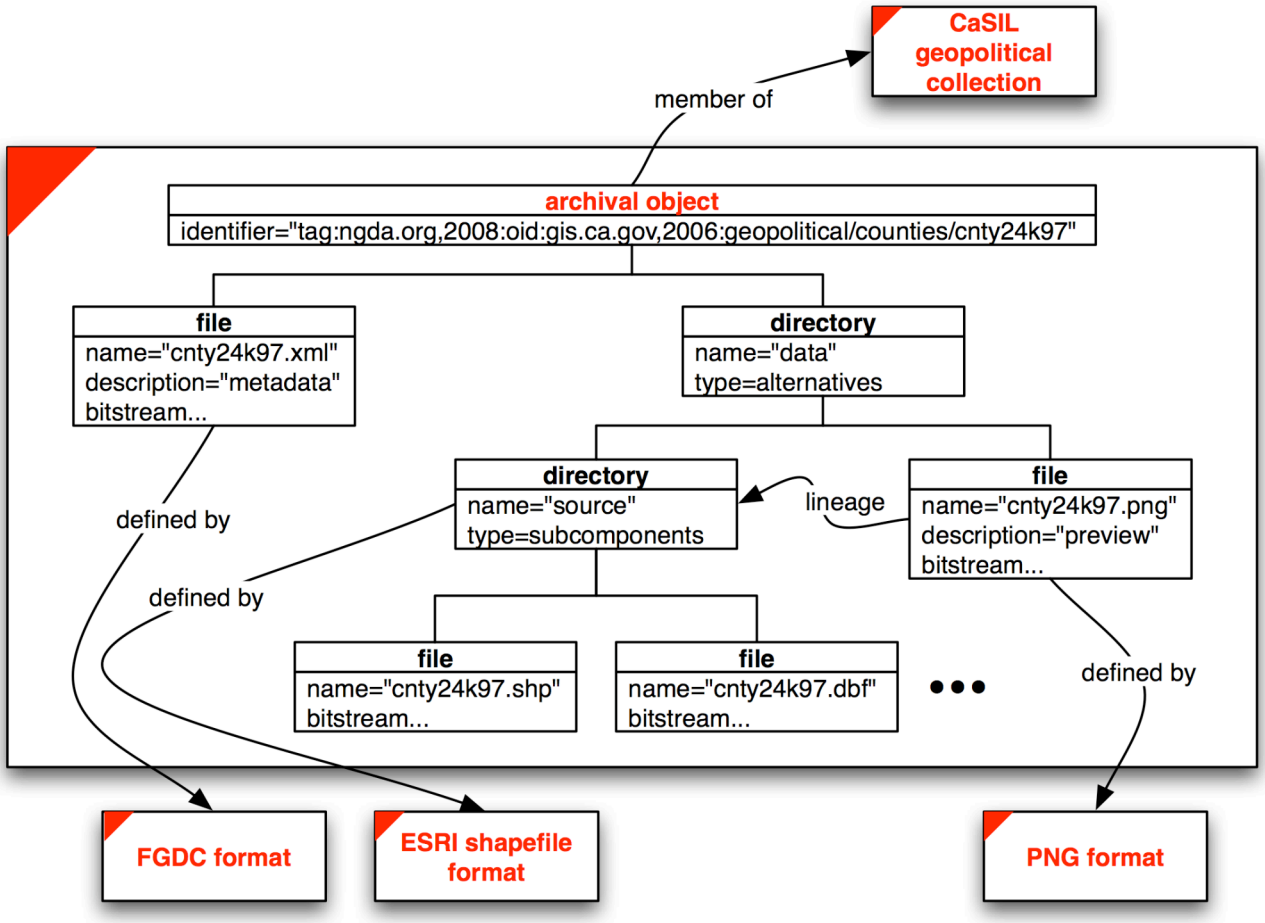

Figure 3. Example archival object (abridged).

represented as a directory tree of files, with physical directories and files corresponding directly to object components. In the root directory of each archival object is an extra file (`manifest.xml`) that encodes the structure of the object and contains archive-model-related metadata such as definitions, relationships, lineage, and fixity information. An example manifest corresponding to the example archival object in Figure 3 is shown in Figure 4. (XML schemas and other documentation on the data model can be viewed at http://www.ngda.org/data-model/.)

Object identifiers are unconstrained, but for a technical reason described below we require that they adhere to URI syntax [3]. An object is stored in the filesystem at the path named by its identifier. Thus an object with identifier ***i*** is stored at path ***i***, the object's manifest is at ***i*****`/manifest.xml`**, and a component at path ***a/b*** within the object is stored at path ***i/a/b***. To avoid performance problems with placing too many files (i.e., too many archival objects) in one filesystem directory, in our implementation we split identifiers into a few pieces so that, for example, identifier ***abcd*** might become pathname ***a/b/c/d***. An explicit, bidirectional mapping from identifiers to pathnames is required anyway, as certain characters must be escaped to avoid interpretation by the filesystem, so additional splitting of identifiers adds no real burden.

In the logical data model, all link destinations are archival objects except lineage links, which may reference objects or components. To unambiguously distinguish these two cases, in links to components the component path is expressed as a URI fragment. For example, a link to component ***a/b*** within object **`tag:ngda.org,2008:`*****i*** would be expressed as **`tag:ngda.org,2008:`*****i*****`#`*****a/b***. It is for this reason only that we constrain object identifiers to be URIs.

Archive self-description can be achieved by storing the manifest schema (which effectively describes the NGDA data model) as an archival object itself and referencing it from manifests via the `xsi:schemaLocation` attribute in the XML header. The fundamental interpretability of the archive thus depends on knowledge of XML, knowledge of the character set(s) in use, and the ability to navigate the filesystem.

This simple physical representation serves several purposes. First, it provides a fallback mechanism: the repository system and higher-level components may no longer be supported and may even have failed, but the content is still accessible as a filesystem. Second, filesystems have proven to be a remarkably resilient concept, and we expect that to continue. And third, storing archival objects as files in native formats makes it easy to provide access to them.

3.2 Storage abstraction

To accommodate the greatest possible variety of storage implementations, and to facilitate handoffs across storage systems, the NGDA architecture places minimal requirements on underlying storage subsystems. The storage subsystem API assumes only that:

- Bitstreams can be created, deleted, read, and written, but not necessarily modified.
- Directories can be created and deleted.
- Bitstreams and directories are named by pathnames, and can be retrieved by pathname.
- The contents of directories can be listed.

These requirements are of course satisfied by any traditional filesystem, but they're also satisfied by other types of storage technologies such as WebDAV[3] servers and the Amazon S3 network storage service.[4]

The requirement that bitstreams and directories be deletable is perhaps not strict, for it might not be needed by an archive that writes once and never makes mistakes, but we have found it to be necessary in practice.

The requirement that directories be listable brings up an interesting point: we need a means of discovering the contents of an archive. Two approaches suggest themselves: either we must assume that the storage system is capable of listing its contents (the approach we have adopted) or we must assume that there is a root object from which all other objects can be discovered by recursive crawling. However, the latter approach requires that aggregate objects such as collections and series maintain "has member" links to their members, which imposes significant performance penalties during large ingest operations.

3.3 Components

The NGDA architecture is defined by its data model and storage abstraction; the components built on top of these may differ from archive to archive to address differing archive needs and content types. In this section we describe the components developed for the prototype archive at UCSB, to demonstrate how an end-to-end archive that provides ingest, search, and access functionality can be built on the architecture.

3.3.1 Archive Server

The Archive Server (Figure 1) acts as a mediator between higher-level components and the archive's underlying storage system. It implements the storage subsystem API, thus insulating other parts of the system from changes in storage technology. Software drivers for different storage technologies can be installed via a plugin software architecture.

All interactions with the server occur through HTTP requests. To perform an ingest operation, a client constructs an `ingest.xml` file containing one or more ingest language commands and submits it along with any pertinent files as a single POST request to the server. Object updates are handled similarly.

As the gateway through which all data must pass before entering the archive, the Archive Server is the ideal place to perform processing tasks such as file format validation and creation of derivative forms (thumbnail images, desiccated versions of poorly supported file formats, etc.). The Archive Server provides hooks to add such processing steps as needed.

[3] http://www.webdav.org/

[4] http://aws.amazon.com/s3

```
<manifest>
  <objectIdentifier> ... </objectIdentifier>
  <relationship name="member of"
     targetObjectRef="..."/>
  <file>
    <name> cnty24k97.xml </name>
    <description> metadata </description>
    <definitionRef> ... </definitionRef>
    <size> ... </size>
    <signature algorithm="MD5"> ... </signature>
  </file>
  <directory type="alternatives">
    <name> data </name>
    <directory type="subcomponents">
      <name> source </name>
      <definitionRef> ... </definitionRef>
      <file>
        <name> cnty24k97.shp </name>
        <size> ... </size>
        <signature algorithm="MD5"> ... </signature>
      </file>
      ...
    </directory>
    <file>
      <name> cnty24k97.png </name>
      <description> preview </description>
      <definitionRef> ... </definitionRef>
      <lineage>
        <sourceComponentRef>
          data/source
        </sourceComponentRef>
      </lineage>
      <size> ... </size>
      <signature algorithm="MD5"> ... </signature>
    </file>
  </directory>
</manifest>
```

Figure 4. Example object manifest (abridged).

3.3.2 Ingest Crawler

Building on the Archive Server's ability to ingest a single archival object at a time, we have developed a bulk ingest system that allows arbitrarily large, homogeneous collections to be ingested with a minimal amount of work. The Ingest Crawler is configured by source, mapping, and target templates that specify which source files are of interest, how the files are to be identified and grouped, what the files' formats are expected to be, and how archival objects are to be packaged. The crawler then works by recursively crawling through the directory structure of a data source.

Archival object identifiers can be created in a variety of ways, including mapping from source filenames or examining and extracting specified fields within known file types.

The software is highly modular, and includes modules for handling a variety of common scenarios, from handling stereotypical file organizations to extracting metadata from relational databases.

3.3.3 *Workflow Tool*

The Workflow Tool is a graphical tool that allows archive administrators to orchestrate ingest runs and perform quality control on the collections assembled by the ingest crawler. Because ingesting large collections may be an extremely long-lived operation due to the significant resources required, the Workflow Tool allows all functions to be checkpointed and restarted. Objects that fail automated processing checks (for example, by lacking a mandatory component as specified by the collection's template) are set aside for manual review. Reports are generated at each step of the workflow detailing the nature of (and possible remedies for) any failures.

3.3.4 *Format Registry*

We initially created a format registry just as a place to store the information we accumulated regarding geospatial and related file formats. Over time, however, the Format Registry has grown to become a collaborative tool that serves as a focal point for collecting format specifications and format metadata from a broad community, and for maintaining, updating, and correcting format entries. The Format Registry is built on a wiki platform in recognition that the expertise of a wide variety of users of formats is a critical element to be captured and archived. The Registry is controlled by archive curators, and includes mechanisms for identifying and authorizing contributors so that access can be limited to those with trusted expertise. Workflow tagging tracks a format entry's maturity. Functions are available to curators to export format entries out of the wiki and ingest them as archival objects in the archive and vice versa.

3.3.5 *Web-based Access*

To provide the simplest and most direct kind of access to the archive, we have placed a web server on top of the archive filesystem. By employing a styling servlet and XSLT transformations that operate on archival object manifests, each archival object appears to web clients as a web page containing hyperlinks to the object's components; an example is shown in Figure 5. The result is that search engines such as Google can spider over an archive's content, making it immediately searchable and accessible down to the level of individual files.

3.3.6 *Alexandria Digital Library*

Given NGDA's focus on geospatial data, which is poorly supported by traditional text search engines, we use the Alexandria Digital Library (ADL) [7] to search over archive content by geographic location, time, type, format, and other criteria, and to provide access mechanisms suitable for geospatial data. Populating ADL with archive content is currently performed by manually-configured and manually-invoked tools, but we are developing a mapping component that will automatically crawl over archive content and map descriptive object metadata to ADL's search indices by employing a registry of known metadata mappings [9].

ADL is a federated system, and supports distributed search over heterogeneous collections spanning multiple nodes. In the current implementation of our archive architecture, ADL provides access-level federation across multiple archives, but as we discuss in section 7, we are working on additional federation mechanisms.

4. IMPLEMENTATION

The prototype archive at UCSB is implemented mainly in Java. The storage subsystem is a 20TB Archivas cluster, which actively detects and repairs corruption. In a nice parallel to the NGDA architecture, Archivas stores all content and its own metadata as simple files in a filesystem, all of which can be recovered (should the Archivas system or company fail) simply by unplugging the disks and plugging them into another machine.

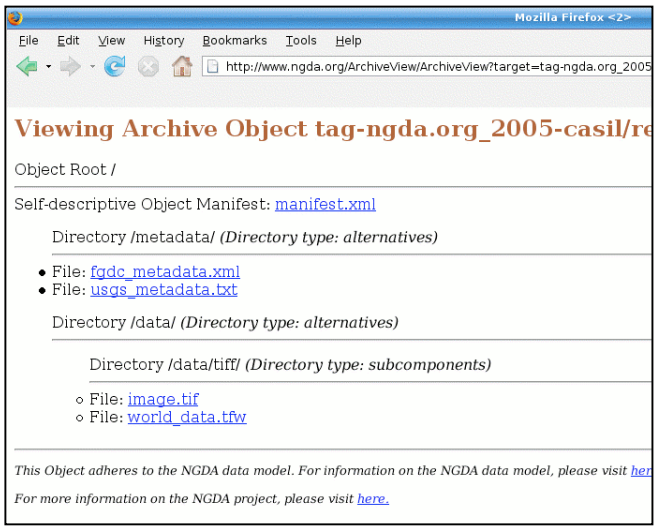

Figure 5. Web view of an archival object.

The Format Registry is based on MediaWiki and PHP. To support the kinds of controlled format entry structure and workflow we desired, we implemented new markup tags, and took advantage of MediaWiki's event triggering system to automatically process and export format entries.

The prototype archive was tested by ingesting several terabytes of data from the California Spatial Information Library (CaSIL).[5] CaSIL is a repository of California's state-owned physical and cultural geospatial data. It has served as an ideal test collection for several reasons. First, the data is both public domain and, since CaSIL has no guaranteed funding, at some risk of loss. Also, CaSIL's mission is content distribution rather than preservation. Most compellingly, CaSIL includes a wide variety of geospatial data. Many different file formats are represented within the data, including multiple image formats (JPEG, MrSID, TIFF) as well as GIS formats (Shapefiles, coverages, etc.).

We have also begun ingesting the David Rumsey Historical Maps collection.

5. RELATED WORK

Like many preservation-related projects, our work on the NGDA architecture was strongly influenced by the Reference Model for an Open Archival Information System (OAIS) [4]. The NGDA data model can be thought of as an instantiation of the conceptual model described by OAIS. To take one example of the relationship and distinction between OAIS and NGDA, OAIS specifies that an archival object's "representation information" is logically contained within the object, but may or may not be physically contained, and that an archive should be able to export an archival object in a "dissemination information package" that may or may not include an unspecified amount of its representation information. NGDA instantiates this conceptual model by specifying logical and physical data models that prescribe how semantics are represented in the archive, how

[5] http://gis.ca.gov/

semantics are related to archival objects, and how they can be accessed in relation to archival objects.

Our work on formats and format representation was largely influenced by early work on the Global Digital Format Registry [1]. NGDA's contribution is to recast the concept of a format registry—often thought of as an entity separate from an archive with its own, separate data model—as an archive in and of itself, and to fully integrate data and formats into a single preservation context.

Additional related work is described in Section 7 below.

6. CONCLUSION

We started the NGDA project with a mature digital library system in hand—the Alexandria Digital Library (ADL)—which hadn't really addressed the issues of long-term preservation. But given the cost of trying to preserve information on a large scale and for a long period of time, we chose not to simply add preservation features onto ADL, and thereby create a kind of ADL++, but instead to start from scratch and ask: What is the minimal approach we can take to mitigate against the risk of permanent loss of information?

The initial result of this work is the set of requirements given in section 2.5. Interestingly, the requirement of providing a fallback representation, and the requirement of facilitating easy handoff, both lead to the same solution: a simple, self-describing data model with a physical representation based on files and filesystems. This solution, while developed for geospatial data, is quite generic and appears to be applicable to many other types of information as well.

In the larger context of the problem of digital preservation, we see the NGDA architecture serving three roles: as a fallback representation that guards against loss of information; as a working representation that supports preservation activities such as active monitoring and format migration; and as a foundation for building end-to-end archives that provide ingest, search, and access functionality.

7. FUTURE WORK

The NGDA architecture is a work in progress, and we are continuing development in a number of directions.

Regarding the data model, there are three areas of work. First, we are adding the ability for objects in one archive to reference objects in another (via definition, lineage, or general relationship links), thus inducing a dependency relationship between the two archives. This addition would allow, for example, multiple archives to reference a central format registry. The result will be an archive-level federation mechanism that complements the access-level federation currently provided by ADL.

Second, we see value in creating a standardized "root" archival object that serves as a kind of whole-archive descriptor. The descriptor would contain archive-level documentation and pointers to the top-level collection objects within the archive, and list key archive dependencies such as any dependent archives and the identifier resolution system(s) used by the archive. The descriptor would also be a natural place to put archive policy declarations such as are being developed by the iRODS project [13].

And third, we would like to express the NGDA data model as a restricted profile of other data models and standards such as the Object Reuse and Exchange (ORE) project's data model [16] and the Metadata Encoding and Transmission Standard (METS) [5]. We believe profiles would be valuable for two reasons. First, to the extent that specifications like ORE and METS are broadly adopted, adopting the relevant parts of their models will serve to relate NGDA's data model, which focuses on long-term preservation specifically, to more general specifications that are more widely understood and more likely to be widely implemented in software systems. Second, NGDA's data model will illustrate which features of the more general specifications are mandatory for long-term preservation, which are merely compatible, and which are incompatible. To look at one example, consider that the ORE data model (as of this writing) supports relationships from objects and object components to formats and other semantic specifications, but does not mandate the existence of any relationships and does not specify the form that format and other semantics specifications must take, in contrast to NGDA's data model which mandates definition links and representation of definitions as archival objects. Similar differences exist between the NGDA data model and the data model implicitly defined by METS. The effect of expressing the NGDA data model as profiles of ORE and METS will be to describe the activity of long-term preservation as an application of these more general technologies.

At the storage level, we are looking at making NGDA's storage abstraction more robust. Currently, NGDA assumes that the extent of an archive is simply the entire contents of the underlying storage system, as opposed to being defined by explicit ownership and control mechanisms. We are currently working on placing the NGDA data model on top of Logistical Networking storage technology [2]. The result will be an architecture that further facilitates archive handoffs.

Lastly, repository systems such as DSpace[6] and Fedora [11] have begun to embrace long-term preservation concerns as well. These systems implement some of the features in NGDA's architecture (storage system abstraction, fallback representation) but not all (integral representation of definitions and other context). We would like to explore the extent to which these repository systems can implement NGDA's architecture. Ultimately, we believe that it is less important to say that data is in DSpace, or any other particular system, and more important to be able to say that the data is in a form that can be easily handed off from whatever system it is in now to the next system that will inevitably come along.

8. ACKNOWLEDGMENTS

The authors would like to thank Catherine Masi of the Map & Imagery Laboratory and David Valentine of SDSC for their contributions to this work.

9. REFERENCES

[1] Stephen L. Abrams (2005). "Establishing a Global Digital Format Registry." *Library Trends* **54**(1) (Summer 2005). http://muse.jhu.edu/journals/library_trends/v054/54.1abrams.pdf

[2] Micah Beck, Terry Moore, James S. Plank, and Martin Swany (2000). "Logistical Networking: Sharing More Than the Wires." In *Active Middleware Services* (Salim Hariri, Craig A. Lee, and Cauligi S. Raghavendra, eds.) (Norwell, Massachusetts: Kluwer Academic Publishers, 2000).

[6] http://www.dspace.org/

[3] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter (2005). *Uniform Resource Identifier (URI): Generic Syntax*. IETF RFC 3986. http://www.ietf.org/rfc/rfc3986.txt

[4] Consultative Committee for Space Data Systems (2002). *Reference Model for an Open Archival Information System (OAIS)*. CCSDS 650.0-B-1, Blue Book (January 2002). http://public.ccsds.org/publications/archive/650x0b1.pdf

[5] Morgan V. Cundiff (2004). "An Introduction to the Metadata Encoding and Transmission Standard (METS)." *Library Hi Tech* **22**(1): 52–64. doi:10.1108/07378830410524495

[6] Margaret Hedstrom (2001). "Exploring the Concept of Temporal Interoperability as a Framework for Digital Preservation." *Third DELOS Workshop on Interoperability and Mediation in Heterogeneous Digital Libraries* (September 8–9, 2001; Darmstadt, Germany). http://www.ercim.org/publication/ws-proceedings/DelNoe03/10.pdf

[7] Greg Janée and James Frew (2002). "The ADEPT Digital Library Architecture." *Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)* (July 14–18, 2002; Portland, Oregon): 342–350. doi:10.1145/544220.544306

[8] Greg Janée, James Frew, and David Valentine (2003). "Content Access Characterization in Digital Libraries." *Proceedings of the 2003 Joint Conference on Digital Libraries (JCDL)* (May 27–31, 2003; Houston, Texas): 261–262. doi:10.1109/JCDL.2003.1204874

[9] Greg Janée and James Frew (2005). "A Hybrid Declarative/Procedural Metadata Mapping Language Based on Python." *Research and Advanced Technology for Digital Libraries: Proceedings of the 9th European Conference (ECDL)* (September 18–23, 2005; Vienna, Austria): 302–313. doi:10.1007/11551362_27

[10] David Cay Johnston (2003). "At I.R.S., a Systems Update Gone Awry." *New York Times*, December 11, 2003. http://www.nytimes.com/2003/12/11/business/11irs.html

[11] Carl Lagoze, Sandy Payette, Edwin Shin, and Chris Wilper (2006). "Fedora: An Architecture for Complex Objects and their Relationships." *International Journal on Digital Libraries* **6**(2) (April 2006): 124–138. doi:10.1007/s00799-005-0130-3

[12] Mike Linda (2006). "OMPS Aggregation and Packaging." *2006 CLASS Users' Workshop* (August 7–8, 2006; Boulder, Colorado). http://ngdc.noaa.gov/dmsp/2nd_class_workshop/class.html

[13] Arcot Rajasekar, Mike Wan, Reagan Moore, and Wayne Schroeder (2006). "A Prototype Rule-based Distributed Data Management System." *HPDC Workshop on Next-Generation Distributed Data Management* (June 20, 2006; Paris, France). http://irods.sdsc.edu/pubs/RODs-paper.doc

[14] Clay Shirky (2005). "AIHT: Conceptual Issues from Practical Tests." *D-Lib Magazine* **11**(12) (December 2005). doi:10.1045/december2005-shirky

[15] Julie Sweetkind-Singer, Mary Lynette Larsgaard, and Tracy Erwin (2006). "Digital Preservation of Geospatial Data." *Library Trends* **55**(2) (Fall 2006). http://muse.jhu.edu/journals/library_trends/v055/55.2sweetkind-singer.pdf

[16] Herbert Van de Sompel and Carl Lagoze (2007). "Interoperability for the Discovery, Use, and Re-Use of Units of Scholarly Communication." *CTWatch Quarterly* **3**(3) (August 2007): 32–41. http://www.ctwatch.org/quarterly/articles/2007/08/interoperability-for-the-discovery-use-and-re-use-of-units-of-scholarly-communication/