Buckets: A New Digital Library Technology for Preserving NASA Research

Journal of Government Information (2001), 28(4), pp. 369-394.

Michael L. Nelson

NASA Langley Research Center
MS 124
Hampton, VA 23681
m.l.nelson@larc.nasa.gov
http://mln.larc.nasa.gov/~mln/
+1 757 864 8511
+1 757 864 8342 (f)

Keywords:
Digital Libraries
Digital Preservation
Intelligent Agents
Scientific and Technical Information
Smart Objects, Dumb Archives
Open Archives Initiative

Buckets: A New Digital Library Technology for Preserving NASA Research

Michael L. Nelson
m.l.nelson@larc.nasa.gov

Abstract

A fundamental task of research organizations is preservation and dissemination of their intellectual output. Historically, this has been accomplished with hard copy formats through a multi-tiered approach of using the open literature, self-publishing, and an array of cooperative libraries and depositories. It can be argued that the historical approach is less than optimal with respect to results achieved and resources expended. However, there have been recent advances in the area of digital libraries (DLs) that address some of the shortcomings of traditional hard copy preservation and dissemination. One of these technologies is "buckets," an aggregative, intelligent construct for publishing. Buckets exist within the "Smart Object, Dumb Archive" (SODA) DL model, which can be summarized as promoting the importance and responsibility of individual information objects and reducing the role of traditional archives and database systems. The goal is that smart objects will be independent of and more resilient to the transient nature of information systems. This paper examines the motivation for buckets and SODA, as well as discussing some initial experiences in using these DL technologies in some U.S. government research laboratories in NASA, the Air Force, and the Department of Energy.

Introduction

Preservation and dissemination of intellectual output and research experiences is a primary concern for all research institutions. However, in practice information preservation is often difficult, expensive and not considered during the information production phase. For example, Henderson (1999) provides data showing for the period of 1960-1995 that "knowledge conservation grew half as much as knowledge output," as a result of research library funding decreasing relative to increasing research and development spending (and a corresponding increase in publications). In short, more information is being produced, and it is being archived and preserved in fewer libraries, with each library having fewer resources. Though eloquent arguments can be presented for the role for and purpose of traditional libraries and data can be presented for the monetary savings libraries can provide (Griffiths & King, 1993), the fact remains that traditional libraries are expensive. Furthermore, the traditional media formats (i.e. paper, magnetic tapes) housed in the traditional libraries are frail, requiring frequent upkeep and are subject to environmental dangers (Lesk, 1997; U.S. GAO, 1990). Digital library (DL) technologies have allowed some commercial publishers to become more involved with library functions, serving on the World Wide Web (WWW) the byproducts of their publishing process (PostScript, PDF, etc.). However, ultimately the goals of publishers and the goals of libraries are not the same, and the long-term commitment of publishers to provide library-quality archival and dissemination services is in doubt (Arms, 1999). While not a panacea, an institution's application of DL technologies will be an integral part of its knowledge usage and preservation effort, in either supplanting or supplementing traditional libraries.

All of this has tremendous impact on a U.S. government agency like NASA. Beyond attention grabbing headlines for its various space programs, NASA ultimately produces

information. The deliverables of NASA's aeronautical and space projects are information for either a targeted set of customers (often industrial partners) or for science and posterity. The information deliverables can have many forms: publications in the open literature; a self-published technical report series; and non-traditional media types such as data and software. NASA contributions to the open literature are subject to the same widening gap in conservation and output identified earlier. For some, the NASA report series is either unknown or hard to obtain (Roper, et al., 1994). For science data, NASA has previously been criticized for poor preservation of this data (U.S. GAO, 1990). However, NASA has identified and is addressing these problems with ambitious goals. From the NASA Scientific and Technical Information (STI) Program Plan:

> By the year 2000, NASA will capture and disseminate all NASA Scientific and Technical Information and provide access to more worldwide mission-related information for its customers. When possible and economical, this information will be provided directly to the desktop in full-text format and will include printed material, electronic documentation, video, audio, multimedia products, photography, work-in-progress, lessons-learned data, research laboratory files, wind tunnel data, metadata, and other information from the scientific and technical communities that will help ensure the competitiveness of U.S. aerospace companies and educational institutions (NASA, 1998).

Although tempered with the phrase "possible and economical," it is clear that the expectations are much higher than simply automating traditional library practices. Much of the STI identified above has historically not been included in traditional library efforts, primarily

because of the mismatch in hard- and soft-copy media formats. However, the ability to now document the entire research process and not just the final results presents entirely new challenges about how to acquire and manage this increased volume of information. To implement the above mandate effectively, additional DL technology is required.

Why Digital Libraries?

A common question regarding digital libraries is "Why not just use existing WWW tools/methods?" Indeed, most DLs use the WWW as the access and transport mechanism. However, it is important to note that while the WWW meets the rapidity requirement of STI dissemination, it has no intrinsic management or archival functions. Just as a random collection of books and serials do not make a traditional library, a random collection of WWW pages does not make a DL. A DL must possess acquisition, management, and maintenance processes. These processes will vary depending on the customers, providers and nature of the DL, but these processes will exist in some format, implicitly or explicitly.

There have been proposals to subvert the traditional publication process with authors self-publishing from their own WWW pages (Harnad, 1997). However, while this availability is useful, pre-prints (or re-prints) linked from a researcher's personal home page are less resilient to changes in computer infrastructure, organization changes, and personnel turnover. Ignoring the socio-political issues of (digital) collegial distribution, there is an archival, or longevity, element to DLs that normal WWW usage does not satisfy. The average lifetime of a uniform resource locator (URL) has been estimated at 44 days (Kahle, 1997), clearly insufficient for traditional archival expectations. Uniform Resource Names (URNs) can be used to address the transient nature of URLs. URNs provide a unique name for a WWW object that can be mapped to a URL

by a URN server. The relationship between URNs and URLs is the same as Internet Protocol (IP) names and IP addresses, respectively. CNRI Handles (Sun & Lannom, 2001), Persistent URLs (Purls) (Shafer, Weibel, Jul & Fausey, 1996) and Digital Object Identifiers (DOIs) (Paskin, 1999)are some common URN implementations. However, no URN implementation has achieved the ubiquity of URL use, and significant maintenance is required to keep a large collection of URNs current. In summary, a DL defines a well-known location for STI to be placed, managed, and accessed. Given the prevalence of the WWW, the well-known location that a DL provides is likely to be WWW accessible.

The Pyramid of STI

NASA communicates its research findings through the traditional open literature process as well as its own multi-tiered, self-published report series (Pinelli, 1990). The NASA report series offers a number of advantages to authors: no page restrictions, potential for restricting dissemination, possibility of color graphics, and occasionally the inclusion of a CD-ROM of data, images or software. However, the latter two are rarer than most authors would like because they are expensive to create, and their distribution is more expensive still. The NASA reports are often ingested in systems that can handle only paper hard copy or possibly just microfiche -- leaving few options for propagation of additional media formats such as CD-ROMs.

An even more compelling case for capturing grey literature at NASA is that the formal publications (NASA's report series or open literature) represent a decreasing percentage of the total amount of STI created and used by NASA and its customers. Due to the increasingly proprietary nature of NASA's work, as well as increasing time constraints on fewer staff members, many research projects are no longer resulting in a formal publication. Instead, the

projects remain as a collection of briefings, data, and other forms of grey literature -- often with unclear or undocumented access restrictions. While neglecting formal publication achieves the short-term goal of increased project turn around time, the inability to capture and preserve the resultant STI creates a gap in the corporate memory. Figure 1 shows the total of number of technical reports, contractor reports, journal articles, conference presentations by Langley Research Center authors from 1991 through 1999. The amount of publications in 1999 is almost half that for 1991.

[figure 1]

These declining numbers are in contrast to the results of Barclay, Pinelli and Kennedy (1997), which found that, at least for the aerospace community, the publications are still in demand and useful, as well as the work of Kaplan and Nelson (2000), which shows that the Langley Research Center DL continues to experience increasing traffic. Research is still being performed, and is still valued by NASA's customers, but there is no well-defined, large-scale publishing outlet for the majority of the currently produced STI. The journal article (or technical report, or whatever the canonical technical publication is in a given environment) is actually an abstract for a much larger body of work (Figure 2). A typical research project summary publication will likely be supported by a host of less formal documents, software, test data, shift-notes, video, images, intermediate analysis, etc. Most traditional and digital libraries focus only on preserving and disseminating the top strata of this pyramid, and the supporting strata, with no processes in place for their preservation, eventually become lost. Furthermore, not all research projects reach the point of producing a canonical publication that can be recorded and tracked by

a traditional or digital library. Because there is no publishing vector for all the strata of the pyramid from research institutions' point of view, valuable information objects are discarded, forgotten or simply lost. Walters and Schockmel (1998) state in regards to previous efforts to manage U.S. government sponsored technical reports: "As for unpublished technical reports, many thousands of these continue to exist beyond the pale of effective bibliographic control, eluding the researchers who could benefit most from them." In fact, their grim picture is not grim enough. Much STI has been lost long before the counting of technical reports begins.

[figure 2]

Information Survivability

The longevity of digital information is a concern that may not be obvious at first glance. While digital information has many advantages over traditional printed media, such as ease of duplication, transmission and storage, digital information suffers unique longevity concerns that hard copy does not, including short life spans of digital media (and their reading devices) and the fluid nature of digital file formats (Rothenberg, 1995; Lesk, 1997). The Task Force on Archiving of Digital Information (1996) distinguished between: *refreshing*, periodically copying the digital information to a new physical media; and *migrating*, updating the information to be compatible with a new hardware/software combination. Refreshing and migrating can be complex issues. The nature of refreshing necessitates a hardware-oriented approach (perhaps with secondary software assistance). Software objects cannot directly address issues such as the lifespan of digital media or availability of hardware systems to interpret and access digital media, but they can implement a migration strategy in the struggle against changing file formats. An

aggregative software object could allow for the long-term accumulation of converted file formats. Rather than successive (and possibly lossy) conversion of:

"Format1 → Format2 → Format3 → …. → FormatN"

One should have the option of:

"Format1 → Format2

Format1 → Format3

Format1 → ….

Format1 → FormatN"

With each intermediate format stored in the same location. This would allow us to implement the "throw away nothing" philosophy, without burdening the DL directly with increasing numbers of formats.

For example, a typical research project at NASA Langley Research Center produces information tuples: raw data, reduced data, manuscripts, notes, software, images, video, etc. Normally, only the report part of this information tuple is officially published and tracked. The report might reference on-line resources, or even include a CD-ROM, but these items are likely to be lost, degrade, or become obsolete over time. Some portions such as software, can go into separate archives (i.e., COSMIC – the official NASA software repository) but this leaves the researcher to locate the various archives, then re-integrate the information tuple by selecting pieces from the different, and perhaps, incompatible archives. Most often, the software and other items, such as datasets are simply discarded or effectively lost in informal, short-lived personal

archives. After 10 years, the manuscript is almost surely the only surviving artifact of the information tuple. As an illustration, COSMIC ceased operation in July 1998; its responsibilities were turned over to NASA's technology transfer centers. However, at the time of this writing there appears to be no operational successor to COSMIC. Unlike their report counterparts in traditional libraries or even DLs, the software contents of COSMIC have been unavailable for several years, if not completely lost.

Additional steps can be taken to insure the survivability of the information object. Data files could be bundled with the application software used to process them if sufficiently common, different versions of the application software, with detailed instructions about the hardware system required to run them, could be a part of the DL. Furthermore, sufficient information could be included to guide the future user in selecting (or developing) the correct hardware emulator.

Buckets

A bucket is a storage unit that contains data and metadata, as well as the methods for accessing both. The motivation for buckets came from experience in the design, implementation and maintenance of NASA scientific and technical information DLs, including the Langley Technical Report Server (LTRS) (Nelson, Gottlich & Bianco, 1994), the NASA Technical Report Server (NTRS) (Nelson, et al., 1995), and the NACA Technical Report Server (NACATRS) (Nelson, 1999). In early user evaluation studies of these DLs, one reoccurring theme was detected. While access to the technical report (or re/pre-print) was desirable, users particularly wanted the raw data collected during the experiments, software used to reduce the data, and ancillary information used in the production of the published report. In response, a

desired goal was to permit DLs to accommodate requests for substantially more information types than just reports. However, rather than create separate DLs for each information type or stretch the definition of a traditional report to include various multi-media formats, a method was sought to define an arbitrary digital object with the capacity to capture and preserve the potentially intricate relationship between multiple information types.

Additionally, the researcher's experiences with updating the NASA DLs and making the content accessible through other DLs and web-crawlers led to the decision to make the information objects intelligent. Objects need to receive maximum exposure to be certain they are not "trapped" inside the DLs, with the only method for their discovery coming from a DL interface. However, the DL should have more than just an exportable description of how to access the objects in the DL. The information object should be independent of the DL, with the capability to exist outside of the DL and move in and out of different DLs in the future. However, to not assume which DL was used to discover and access the buckets means that the buckets must be self-sufficient and perform whatever tasks are required of them, potentially without the benefit of being arrived at through a specific DL.

The bucket design goals are: aggregation, intelligence, self-sufficiency, mobility, heterogeneity and archive independence. It is difficult to overstress the importance of aggregation as a design goal. In experience with other NASA DLs, data was often partitioned by its semantic or syntactic type: metadata in one location, PostScript files in another location, PDF files in still another location, etc. Over time, different forms of metadata were introduced for different purposes, the number of available file formats increased, the services defined on the data increased, new information types (software, multimedia) were introduced, the logging of actions performed on the objects became more difficult. The result of a report being "in the DL"

eventually represented so much DL jetsam - bits and pieces physically and logically strewn across the system. We responded to this situation with extreme aggregation. As suggested by the SODA model, buckets have unique requirements due to their emphasis on minimizing dependence on specific DL implementations.

The first focus of the aggregation was for the various data types. Based on experience gained while designing, implementing and maintaining LTRS and NTRS, researchers initially decided on a two-level structure within buckets:

- buckets contain 0 or more *packages*
- packages contain 0 or more *elements*

Actual data objects are stored as elements, and elements are grouped together in packages within a bucket. In LTRS and NTRS, a two-level architecture was sufficient for most applications, so this two-level architecture was retained as a simplifying assumption during bucket implementation. Future work will implement the semantics for describing arbitrarily complex, multi-level data objects.

An element can be a "pointer" to another object such as another bucket, or any other arbitrary network object. By having an element "point" to other buckets, buckets can logically contain other buckets. Although buckets provide the mechanism for both internal and external storage, buckets have less control over elements that lie physically outside the bucket. However, it is left to the bucket creator as to the appropriateness of including pointers in an archival unit. Buckets have no predefined size limitation, either in terms of storage capacity, or in terms of number of packages or elements. Buckets are accessed through one or more URLs. For an

example of how a single bucket can be accessed through multiple URLs, consider two hosts that share a file system:

```
http://host1.foo.edu/bar/bucket-27/
http://host2.foo.edu/bar/bucket-27/
```

Both of these URLs point to the same bucket, even though they are accessed through different hosts. Also, consider a host that runs multiple http servers:

```
http://host1.foo.edu/bar/bucket-27/
http://host1.foo.edu:8080/bucket-27/
```

If the http server running on port 8080 defines its document root to be the directory "bar," then the two URLs point to the same bucket.

Elements and packages have no predefined semantics associated with them. Authors can model whatever application domain they desire using the basic structures of packages and elements. One possible model for bucket, package, and element definition is based on NASA DL experiences. In NASA DL buckets, packages represent semantic types (manuscript, software, test data, etc.) and elements represent syntactic representations of the packages (a `.ps` version, `.pdf` version, `.dvi` version, etc.). Other bucket models using elements and packages are possible. For example, we have used buckets for entire research projects and university classes as well as for STI publications.

Buckets provide mechanism, not policy. Yet, buckets have the capability of implementing different policies as well: one site might allow authors to modify the buckets after

publishing, and another site might have buckets be "frozen" upon publication. Still another site might define a portion of the bucket to receive annotations, review, or contributions from the users, while keeping another portion of the bucket frozen, or only changeable by authors or administrators.

Another focus of aggregation was including the metadata with data. Through experience NASA researchers found that metadata tended to "drift" over time, becoming decoupled from the data it described or "locked" in specific DL systems and hard to extract or share with other systems. For some information types such as reports, regenerating lost metadata is possible either automatically or by inspection. For other information types such as experimental data, the metadata cannot be recovered from the data. Once the metadata is lost, the data itself becomes useless. The researcher did not wish to take a proscriptive stance on metadata. Although ultimately the bucket itself has to choose one metadata format as canonical for structural purposes, buckets need to be able to accommodate multiple metadata formats. Buckets do this by storing metadata in a reserved package and using methods for reading and uploading new metadata formats as elements in the metadata package. As a result, buckets can accommodate any number of past, present or future metadata formats.

The final aggregation focus was on the services defined on buckets and the results of those services. The default state is for everything to be contained within the bucket to permit it to display, disseminate, and manage its contents. This includes the source code for all methods defined on the bucket, the user ids and passwords, the access control lists, the logs of actions taken on the bucket, Multipurpose Internet Mail Extensions (MIME) (Borenstein & Freed, 1993) definitions and all other supporting technologies necessary for the bucket to function. The self-

sufficiency and mobility design goals dictate that a bucket cannot make many assumptions about the environment in which it will reside and should require no server modifications to function.

The buckets described here are version 1.6.3. Buckets are currently written in Perl 5 (Wall, Christiansen & Schwartz, 1996) and use http as the transport protocol. However, buckets can be written in any language as long as the bucket API is preserved. Buckets were originally deployed in the NCSTRL+ project (Nelson, Maly, Shen & Zubair, 1998), which demonstrated a modified version of the Dienst protocol (Davis & Lagoze, 2000). Owing to their Dienst-related heritage, bucket metadata is stored in RFC-1807 format (Lasher & Cohen, 1995), with package and element information stored in NCSTRL+ defined optional and repeatable fields. Although buckets use RFC-1807 as their native format, they can contain and serve any metadata type. Dienst has all of a document's files gathered into a single Unix directory. A bucket follows the same model and has all relevant files collected together using directories from file system semantics. The bucket is accessible through a common gateway interface (CGI) script that enforces terms and conditions, and negotiates presentation to the WWW client.

Buckets take advantage of the package/element construct for their internal configuration. In addition to the user data entered as packages and elements, the bucket keeps its own files as elements in certain reserved packages. Thus, methods such as “add_element”, “delete_element” and so forth can be used to update the source code for the bucket, update the password files, etc. Table 1 lists the predefined packages and some of the elements they contain. By convention, these packages begin with an underscore (“_”) character. Figure 3 provides a model representation of the structure of a typical bucket, with internal packages and elements on the left and user-supplied data packages on the right.

[figure 3]

[table 1]

Aside from Perl 5, http, and CGI, buckets make no assumptions about the environment in which they will run. Mobility is one of the design goals of buckets, and a corollary of that is that buckets should not require changes in a "reasonable" http server setup; where "reasonable" is defined to be allowance of the `index.cgi` convention. Once these assumptions have been met, buckets, by default, take care of everything themselves with no server intervention, including MIME typing, terms and conditions, and support libraries. Although bucket development was conducted under Solaris (Unix), buckets have been tested on a variety of Unix and Windows NT systems.

The NASA reference implementation implements the bucket API using http encoding of messages. Buckets appear as ordinary URLs and casual users should not realize they are not interacting with a typical web site. Although possible, users are not expected to invoke methods directly – the applicable methods for accessing its contents are automatically built into the bucket's HTML output. The other creation and management-oriented methods are expected to be accessed by various bucket tools. If no method is invoked via URL arguments, the "display" method is assumed by default. This generates a human-readable display of the bucket's contents. For example, a bucket version of a NACA Technical Note:

```
http://www.cs.odu.edu/~nelso_m/naca-tn-2509/
```

which is the same as:

```
http://www.cs.odu.edu/~nelso_m/naca-tn-2509/?method=display
```

[figure 4]

Both URLs produce the output in Figure 3. These URLs could be reached through either a searching or browsing function within a DL, or typed in directly – buckets make no assumptions about their discovery. However, a DL can pass in preferences to alter the appearance of the bucket. For example, a view (Figure 4) of the bucket suitable for library staff can be specified with:

```
http://www.cs.odu.edu/~nelso_m/naca-tn-
2509/?method=display&
view=staff
```

[figure 5]

From the human readable interface the "display" method generates, if users wish to retrieve the PDF file they click on the PDF link that was automatically generated:

```
http://www.cs.odu.edu/~nelso_m/naca-tn-
2509/?method=display&
pkg_name=report.pkg&element_name=naca-tn-2509.pdf
```

this would cause the WWW browser to launch the PDF reader. Similarly, if users wished to display the scanned pages, selecting the automatically created link would send the following arguments to the "display" method:

```
http://www.cs.odu.edu/~nelso_m/naca-tn-
2509/?method=display&
pkg_name=report.pkg&element_name=report.scan
```

this would produce the output in Figure 5. To the casual observer, the bucket API is transparent. However, if individual users or harvesting robots know a particular URL is a bucket, they can exploit this knowledge. For example, to extract the metadata in default (RFC-1807) format, the URL is:

```
http://www.cs.odu.edu/~nelso_m/naca-tn-
2509/?method=metadata
```

[figure 6]

this would return the metadata in a structured format, suitable for inclusion in an index being automatically built by a DL. If a specific metadata format was desired, it could be requested:

```
http://www.cs.odu.edu/~nelso_m/naca-tn-2509/
?method=metadata&format=marc
```

This will result in a MARC record being returned if the bucket has the metadata in MARC format, or can have it converted into MARC format (see the next section). If a user or agent

wishes to determine that nature of a bucket, a number of methods are available. For example, to determine the bucket's version one would enter:

```
http://www.cs.odu.edu/~nelso_m/naca-tn-2509/?method=version
```

To see what methods are defined on a bucket one would enter:

```
http://www.cs.odu.edu/~nelso_m/naca-tn-
2509/?method=list_methods
```

Even if a harvester is not bucket-aware, it can still "crawl" or "spider" the bucket URLs as normal URLs, extracting information from the HTML human-readable interface generated by the "display" method (assuming the "display" method is not restricted by T&C). Buckets offer many expressive options to the users or services that are bucket-aware, but are transparent to those who are not bucket-aware. All bucket methods are listed in Table 1, and a full discussion of the methods and their arguments can be found in Nelson (2000).

[Table 2]

Bucket Communication Space

Linda, the parallel communication library (Carrriero & Gelernter, 1989), partially motivated the Bucket Communication Space (BCS). In Linda, processes effectively pass messages by creating "tuples" that exist in "tuple space." These data objects are created with the "eval" primitive, and filled with data by processes using the "out" primitive. Processes use "rd"

and “in” for reading and reading-removing operations, respectively. These primitives allow processes to communicate through tuple space, without having to know the details (e.g. hostnames, port numbers) of the processes. The messages written to tuple space can have regular expressions and control logic to specify who should read them. When an “in” tuple sees an “out” tuple and the conditions of the former match that of the latter, the message is communicated to the receiving process and the tuple is removed from tuple space. Though the Linda environment imposes a performance overhead, it provides a useful layer of abstraction for inter-process communication.

Something similar was desired for buckets: buckets communicating with other buckets without having to know the details of bucket location. This is especially important if the buckets are mobile, and a bucket’s location is not guaranteed to be static. The BCS also provides a method for centralizing functionality that cannot be replicated in individual buckets. This could be either because of efficiency concerns (the resulting bucket would be too bloated) or implementation limitations (a service is not available on the architecture that is serving the bucket). Buckets need only know how to communicate to a BCS server, which can handle their requests for them. The location of a BCS server is stored as a preference within the bucket.

The BCS model opens up many possible service areas. A subtle element of the BCS is that buckets, not people, are responsible for the provision and coordination of these services. Proof-of-concept implementations are provided for four significant services: file format conversion, metadata conversion, bucket messaging, and bucket matching. Digital videos of the operation of these services can be found in (Nelson, 2000).

File Format Conversion

File format conversion provides bi-directional conversion of image (e.g. GIF, JPEG) formats and page description formats (e.g., PostScript, PDF). Format conversion is an obvious application – additional formats will become available after a bucket's publication and the ability to either place them in the bucket or dynamically create them will be useful in information migration.

Metadata Conversion

Metadata conversion is similar to file format conversion, providing conversion between some of the more popular metadata formats (e.g., Refer, RFC-1807, bibtex). Metadata conversion is extremely important because although buckets ultimately have to choose a single format to operate on, it is unreasonable to assume that all applications needing metadata from the bucket should have to choose the same format. Being able to specify the desired format to receive from a bucket also leaves the bucket free to change its canonical format in the future.

Bucket Messaging

Messaging allows multiple buckets to receive a message if they match specific criteria. While point-to-point communication between buckets is always possible, bucket messaging provides a method for discovering and then sending messages to buckets. Messaging provides functionality closer to the original inspiration of Linda, and can be used as the core of a "bucket-multicasting" service that sends pre-defined messages to a subset of registered buckets. This could be used in turn to implement a metadata normalization and correction service, such as that described by French, Powell, Schulman and Pfaltz (1997) or Lawrence, Bollacker and Giles (1999).

Bucket Matching

The most compelling demonstration of the BCS is bucket matching. Matching provides the capability to create linkages between "similar" buckets. Consider a technical report published by the Old Dominion University computer science department that is also submitted to a conference. The report exists on the department's DL and the publishing authority is: ncstrl.odu_cs. If the conference paper is accepted, it will eventually be published by the conference sponsor. For example, say the conference sponsor is the Association for Computing Machinery, whose publishing authority would be ncstrl.acm. Although the conference paper will surely appear in a modified format (edited and perhaps abbreviated), the technical report and the conference paper are clearly related, despite being separated by publishing authority, date of publication, and editorial revisions. Two separate but related objects now exist, and are likely to continue to exist.

How best to create the desired linkage between the two objects? It is easy to assume ncstrl.acm has neither the resources nor the interest to spend the time searching for previous versions of a manuscript. Similarly, ncstlrl.odu_cs cannot link to the conference bucket at the creation time of the technical report bucket, since the conference bucket did not exist then. It is unrealistic to suggest the relevant parties will go back to the ncstrl.odu_cs archive and create linkages to the ncstrl.acm bucket after six months to a year have passed. However, if both buckets are registered in the same bucket communication space (by way of sending their metadata or fulltext), they can "find each other" without human intervention. When a match, or near match (the threshold for "match" being a configurable parameter) is found, the buckets can either automatically link to each other, or inform a human reviewer that a potential match has been found and request approval for the linkage.

This technique could also be used to find related work from different authors and even duplications (accidental or plagarious). In the test runs using the National Advisory Committee for Aeronautics (NACA) portion of the Universal Preprint Service (see next section), multi-part reports were found (e.g. Part 1, Part 2) and matched, as were Technical Notes (archival equivalent of a computer science technical report) that were eventually published as Reports (archival equivalent of a journal article), and a handful of errors where duplicate metadata was erroneously associated with multiple reports.

Smart Objects, Dumb Archives

Buckets are part of the larger "Smart Object, Dumb Archive" DL Model (Maly, Nelson & Zubair, 1999). SODA is a reaction to the vertically integrated (and non-interoperable) DLs that tended to grow from the ad-hoc origins of many popular DLs (Esler & Nelson, 1998). Separating the functionality of the archive from that of the DL allows for greater interoperability and federation of DLs. The archive's purpose is to provide DLs the location of buckets (the DLs can poll the buckets themselves for their metadata), and the DLs build their own indexes. If a bucket does not "want" to share its metadata (or contents) with certain DLs or users, its terms and conditions will prevent this from occurring. For example, it is expected that the NASA digital publishing model will begin with technical publications, after passing through their respective internal approval processes, to be placed in a NASA archive. The NASA DL (which is the set of the NASA buckets, the NASA archive(s), and the user communities at each level) would poll this archive to learn the location of buckets published within the last week. The NASA DL could then contact those buckets, requesting their metadata. Other DLs could index NASA holdings in a similar way -- polling the NASA archive and contacting the appropriate

buckets. The buckets would still be stored at NASA, but they could be indexed by any number of DLs, each with the possibility for novel and unique methods for searching or browsing. In another scenario a DL might collect relevant metadata, perform additional filtering and then determine applicability for inclusion into their DL. In addition to an archive's holdings being represented in many DLs, a DL could contain the holdings of many archives. If all digitally available publications are viewed as a universal corpus, then this corpus could be represented in N archives and M DLs, with each DL customized in function and holdings to the needs of its user base.

Just as buckets are a possible implementation of smart objects, there are also different possible implementations of dumb archives. Originally, a separate protocol for archives was defined and implemented as "DA" (Nelson, 2000) but this implementation is no longer being developed. Instead, the DA functionality is now provided by the evolving Open Archives Initiative (OAI) and its metadata harvesting protocol. Similar to the DA, OAI archives for NASA DLs are implemented as modified buckets. These OAI buckets have all the functionality of regular buckets, plus the source code to implement the six verbs of OAI metadata harvesting protocol.

The OAI metadata harvesting protocol defines six “verbs” (Table 2) that allow the creators of DLs (known as “service providers” in OAI parlance) to query archives (“data providers”) to determine the nature of the archive and produce full or partial dumps of an archive’s metadata (Van de Sompel & Lagoze, 2000). Most of the six verbs take various arguments such as date stamps or archive-defined sets to allow for partial harvesting. Although any metadata format can be provided by a data provider, in the interest of easing the task of

creating service providers, the Dublin Core (Weibel, Kunze, Lagoze & Wolfe, 1998) is defined as the default metadata format required for OAI compliance.

[Table 3]

The OAI protocol is not defined as a stand-alone system; only a layer for retrieving items is available. An OAI interface is always a front-end to some other archival system, i.e., a relational database management system, directory service, filesystem, or Dienst server. The goal of the OAI is to provide a standard mechanism for a DL to expose its metadata to external harvesters. This will allow the creation of value added DLs that provide resource discovery to content from multiple archives. Just as buckets break the dependency of the information objects on the archives, the OAI breaks the dependency of archives on the DLs.

The OAI grew out of the meeting surrounding the presentation of the Universal Preprint Service (UPS) demonstration digital library. The UPS is a large DL testbed introduced in October 1999 and is based on NCSTRL+ software. The UPS prototype was a feasibility study for the creation of cross-archive end-user services. With the premise that users would prefer to have access to a federation of digital libraries, the primary purpose of the project was the identification of key issues in actually creating an experimental end-user service for data originating from important existing, production archives. This included a total of almost 200,000 buckets harvested from six existing production DLs (Van de Sompel, et al., 2000).

Future Work

The lessons learned from implementing buckets and the supporting technology, along with their success and popularity in NCSTRL+ and UPS, point to many areas of future work, of

both practical and academic interest. Though buckets have been deployed in a number of testbed DLs, there is currently a project planned for bucket usage among NASA, the Air Force Research Laboratory and Los Alamos National Laboratory to use buckets and the OAI for DL interoperability. Buckets are especially suited for the multiplicity of files and formats resulting from older technical reports that must be scanned.

Alternate Implementations

Although Perl and CGI are good development platforms, other bucket implementations should be explored. This includes making the bucket API available through non-http environments, such as CORBA (Vinoski, 1997) and implementing buckets using other languages and relational database management systems.

Pre-defined Packages and Elements

Some functionality improvements could be made not through new or modified methods, but through conventions established on the current infrastructure. One convention already adopted was the use of a `BCS_Similarity.pkg` package to hold the resulting links of the BCS similarity indexing. Other possible uses include: standard element names for bucket checksums (entire bucket, packages or elements) to insure the integrity of elements; standard packages (or elements) for bibliographic citation information, possibly in multiple encodings; or standard package or element names for previous revisions of bucket material. Conventions are likely to be adopted as need and applications arise.

Increased Intelligence

There are a number of functions for which buckets already have hooks in place, but have not yet been fully automated. For example, the "lint" method can detect internal errors and misconfigurations in the bucket, but it does not yet attempt to repair a damaged bucket.

Similarly, a bucket preference could control the automatic updating of buckets when new releases are available, while still maintaining the bucket's own configuration and local modifications. The updated bucket could then be tested for correct functionality, and rolled back to a previous version if testing fails. The option of removing people from the bucket update cycle would ease a traditional administration burden.

Buckets could also be actively involved in their own replication and migration, as opposed to waiting for human intervention for direction. Buckets could copy themselves to new physical locations so they could survive physical media failures, existing either as functioning or dormant replicates. Should the canonical bucket be "lost" somehow, buckets could vote among themselves to establish a new priority hierarchy. Distributed storage projects such as the Archival Intermemory (Goldber & Yianilos, 1998) or the Internet2 Distributed Storage Infrastructure Project (Beck & Moore, 1998) could serve as complementary technologies for implementing migratory buckets.

Security, Authentication and Terms & Conditions

While every effort has been made to make buckets as secure and safe as possible, a full–scale investigation by an independent party has not been performed. A first level of investigation would be in attacking the buckets themselves to determine if the buckets could be damaged, made to perform actions prohibited by their terms and conditions (T&C) files, or otherwise be compromised. A second level of investigation would be examining if buckets could be compromised through side effects resulting from attacks on other services. Currently, buckets have no line of defense if the web server or the system software itself is attacked. Having buckets employ some sort of encryption on their files that is decoded dynamically would

offer a second level of security, making the buckets truly opaque data objects that could withstand at least some level of attack if the system software was compromised.

Authentication is currently done through the standard http procedures. Authentication alternatives using Kerberos (Steiner, Neuman & Schiller, 1988) MD5 (Rivest, 1992), or X.509 (CCIT, 1998) should be explored so buckets can fit into a variety of large-scale authentication schemes in use at various facilities.

Discipline-Specific Buckets

Buckets are currently not specific to any discipline; they have a generic "one-size-fits-all" approach. While this is attractive for the first generation of buckets since it excludes no disciplines, it also does nothing to exploit assumptions and extended features of a specific discipline. Intuitively, an earth science bucket could have different requirements and features than a computational science bucket. Given a scientific discipline, it could be possible to define special data structures and even special methods or method arguments for the data, such as geo-spatial arguments retrieving data from earth-science buckets or compilation services for a computational science bucket.

Usage Analysis

There are several DL projects that focus on determining the usage patterns of their holdings and dynamically arranging the relationships within the DL holdings based on these patterns (Bollen & Heylighen, 1997; Rocha, 1999). All of these projects are similar in that they extract usage patterns of passive documents, either examining the log files of the DL, or instrumenting the interface to the DL to monitor user activity, or some hybrid of these approaches. An approach that has not been tried is for the objects themselves to participate in determining the usage patterns, perhaps working in conjunction with monitors and log files.

Since the buckets are executable code, it is possible to not just instrument the resource discovery mechanisms, but the archived objects also. NASA has experience instrumenting buckets to extract additional usage characteristics, but has not combined this strategy with that of the other projects.

Software Reuse

Buckets could have an impact in the area of software reuse as well. If a bucket stores code, such as a solver routine, it would not have to be limited to a model where users extract the code and link it into their application. Rather, the bucket could provide the service, and be accessible through remote procedure call (RPC)–like semantics. Interfaces between distributed computing managers such as Netsolve (Casanova & Dongarra, 1998) or NEOS (Czyzk, Mesnier & Moore, 1998) and "solver buckets" could be built, providing simple access to the solver buckets from running programs. Data, and the routines to derive and manipulate it, could reside in the same bucket in a DL. This would likely be tied with a discipline specific application, such as a bucket having a large satellite image and a method for dynamically partitioning and disseminating portions of the data.

Alternatively, users could temporarily upload data sets into the bucket to take advantage of a specialized solver resident within the bucket without having to link it into their own program. This would be especially helpful if the solver had different system requirements, and it could not easily be hosted on a user's own machine. However, the traditional model of "data resides in the library; analysis and manipulation occurs outside the library" can be circumvented by making the archived objects also be computational objects.

Related Work

There are projects with similar aggregation goals as buckets from the DL community, such as Multivalent Documents (Phelps & Wilensky, 2000) and the Kahn-Wilensky Framework (Kahn & Wilensky, 1995) and its derivatives (Warwick Framework (Lagoze, Lynch & Daniel, 1996) and FEDORA (Payette & Lagoze, 2000). Some projects, such as the VERS Encapsulated Objects (VEOs) of the Victorian Electronic Record Strategy (VERS) (Waugh, Wilkinson, Hills & Dellóro, 2000), focus primarily on digital preservation goals. None of these other projects, however, feature mobility, self-sufficiency or the SODA-inspired motivation of freeing the information object from archival control and dependency. Most DL intelligent agent projects focus on aids to the DL user or creator; the intelligence is machine-to-human based. Buckets remain unique because the information objects themselves are intelligent, providing machine-to-machine (or, bucket-to-bucket) intelligence.

Conclusions

Buckets were born of the NASA experience in creating, populating and maintaining several production digital libraries. The users of NASA DLs repeatedly wanted access to data types beyond that of the technical publication. The traditional publication systems and the digital systems that automated them were unable to address their needs adequately. Instead of creating a raft of competing, "separate-but-equal" DLs to contain the various information types, a container object was created capable of capturing and preserving the relationship between or among any number of arbitrary data types.

Buckets are aggregative, intelligent, WWW-accessible digital objects that are optimized for publishing in DLs. Buckets implement the philosophy that information itself is more

important than the DL systems used to store and access information. Buckets are designed to imbue the information objects with certain responsibilities, such as the display, dissemination, protection and maintenance of its contents. As such, buckets should be able to work with many DL systems simultaneously, and minimize or eliminate the necessary modification of DL systems to work with buckets. Ideally, buckets should work with everything and break nothing. This philosophy is formalized in the SODA (Smart Object, Dumb Archive) DL model. The objects become "smarter" at the expense of the archives (that become "dumber"), as functionalities generally associated with archives are moved into the data objects themselves. This shift in responsibilities from the archive into the buckets results in a greater storage and administration overhead, but these overheads are small in comparison to the great flexibility that buckets bring to DLs. Freeing the information objects from the dependency of specific archive software, databases or search engines should increase their chances at long-term survivability.

Buckets are already having a significant impact in how NASA and other organizations such as the Los Alamos National Laboratory, and the Air Force Research Laboratory are designing their next generation DLs. The interest in buckets has been high, and every feature introduced seems to raise several additional areas of investigation for new features and applications. First and most important, the creation of high quality tools for bucket creation, management and maintenance in a variety of application scenarios is absolutely necessary. Without tools, buckets will not be widely adopted. Other short-term areas of investigation include optimized buckets, alternate implementations of buckets, discipline-specific buckets, and extending authentication support to include a wider variety of technologies. Long-range plans include significant utilization of bucket mobility and bucket intelligence, including additional features in the Bucket Communication Space. Buckets, through aggregation, intelligence,

mobility, self-sufficiency, and heterogeneity, provide the infrastructure for information object independence. The truly significant applications of this new breed of information objects remain undiscovered.

References

Arms, W. A. (1999). Preservation of scientific serials: three current examples. Journal of Electronic Publishing, 5(2). Available at http://www.press.umich.edu/jep/05-02/arms.html.

Barclay, R. O., Pinelli, T. E., & Kennedy, J. M. (1997). The Role of the U.S. GovernmentTechnical Report in Aerospace Knowledge Diffusion. In Pinelli, T. E., Barclay,R. O., Kennedy, J. M., & Bishop, A. P. (Eds.). Knowledge Diffusion in the U.S. Aerospace Industry (pp. 707-759). Greenwhich, CT.: Ablex Publishing Corportation.

Beck, M. & Moore, T. (1998). The Internet2 distributed storage infrastructure project: an architecture for Internet content channels. Computer Networking and ISDN Systems, 30(22-23), 2141-2148. Available at http://dsi.internet2.edu/pdf-docs/i2-chan-pub.pdf.

Bollen, J. & Heylighen F. (1997). Dynamic and adaptive structuring of the World Wide Web based on user navigation patterns. Proceedings of the Flexible Hypertext Workshop (pp. 13-17), Southhampton, UK. Available at http://www.c3.lanl.gov/~jbollen/pubs/Bollen97.htm.

Borenstein, N. & Freed, N. (1993). MIME (multipurpose Internet mail extensions) part one: mechanisms for specifying and describing the format of Internet message bodies. Internet RFC-1521. Available at ftp://ftp.isi.edu/in-notes/rfc1521.txt.

Carriero, N. & Gelernter, D. (1989). Linda in context. Communications of the ACM, 32(4), 444-458.

Casanova, H. & Dongarra, J. (1998). Applying Netsolve's network-enabled solver. IEEE Computational Science & Engineering, 5(3), pp. 57-67.

CCITT (1998). The directory authentication framework. CCITT Recommendation X.509.

Czyzyk, J., Mesnier, M. P. & More, J. J. (1998). The NEOS solver. IEEE Computational Science & Engineering, 5(3), pp. 68-75.

Davis, J. R. & Lagoze, C. (2000). "NCSTRL: design and deployment of a globally distributeddigital library." Journal of the American Society for Information Science, 51(3), 273-280.

Esler, S. L. & Nelson, M. L. (1998). Evolution of scientific and technical information distribution. Journal of the American Society for Information Science, 49(1), 82-91. Available at http://techreports.larc.nasa.gov/ltrs/PDF/1998/jp/NASA-98-jasis-sle.pdf.

French, J. C., Powell, A. L., Schulman, E. & Pfaltz, J. L. (1997). Automating the construction of authority files in digital libraries: a case study. In C. Peters & C. Thanos (eds.), Research and advanced technology for digital libraries, first European conference, ECDL '97 (pp. 55-71), Berlin: Springer.

Goldberg, A. V. & Yianilos, P. N. (1998). Towards an archival intermemory. Proceedings of the IEEE forum on research and technology advances in digital libraries (pp. 147-156), Santa Barbara, CA.

Griffiths, J.-M. & King, D. W. (1993). Special libraries: increasing the information edge. Washington, DC: Special Libraries Association.

Harnad, S. (1997). How to fast-forward serials to the inevitable and the optimal for scholars and scientists. Serials Librarian, 30, 73-81. Available at http://www.cogsci.soton.ac.uk/~harnad/Papers/Harnad/harnad97.learned.serials.html.

Henderson, A. (1999). Information science and information policy: the use of constant dollars and other indicators to manage research investments. Journal of the American Society for Information Science, 50(4), 366-379.

Kahle, B. (1997). Preserving the Internet. Scientific American, 264(3).

Kahn, R. & Wilensky, R. (1995) A framework for distributed digital object services. cnri.dlib/tn95-01. Available at http://www.cnri.reston.va.us/home/cstr/arch/k-w.html.

Kaplan, N. R. & Nelson, M. L. (2000). Determining the Publication Impact of a Digital Library. Journal of the American Society for Information Science, 51(4), 324-339.

Lagoze, C., Lynch C. A., & Daniel, R. (1996). The Warwick framework: a container architecture for aggregating sets of metadata. Cornell University Computer Science Technical Report TR-96-1593. Available at http://ncstrl.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.cornell/TR96-1593.

Lasher, R. & Cohen, D. (1995). A format for bibliographic records. Internet RFC-1807. Available at ftp://ftp.isi.edu/in-notes/rfc1807.txt.

Lawrence, S., Bollacker, K. & Giles, C. L. (1999). Distributed error correction. Proceedings of the fourth ACM conference on digital libraries (p. 232), Berkeley, CA.

Lawrence, S. & Giles, C. L. (1998). Searching the World Wide Web. Science, 280, 98-100. Available at http://www.neci.nj.nec.com/~lawrence/science98.html.

Lesk, M. E. (1997). Practical digital libraries: books, bytes & bucks. San Francisco, CA: Morgan-Kaufmann Publishers.

Machie, H. B. & Stewart, S. H. (1999). Scientific and technical information of the Langley Research Center for calendar year 1998. NASA Technical Memorandum NASA/TM-1999-209095. Available at http://techreports.larc.nasa.gov/ltrs/PDF/1999/tm/NASA-99-tm209095.pdf

Maly, K., Nelson, M. L., & Zubair, M. (1999). Smart objects, dumb archives: a user-centric, layered digital library framework. D-Lib Magazine, 5(3). Available at http://www.dlib.org/dlib/march99/maly/03maly.html.

NASA (1998). NASA Scientific and Technical Information (STI) program plan. Available at http://stipo.larc.nasa.gov/splan/

Nelson, M. L., Gottlich, G. L., & Bianco, D. J. (1994). World Wide Web implementation of the Langley technical report server. NASA TM-109162. Available at http://techreports.larc.nasa.gov/ltrs/PDF/tm109162.pdf.

Nelson, M. L., Gottlich, G. L., Bianco, D. J., Paulson, S. S., Binkley, R. L., Kellogg, Y. D., Beaumont, C. J., Schmunk, R. B., Kurtz, M. J., Accomazzi, A., & Syed, O. (1995). The NASA technical report server. Internet Research: Electronic Network Applications and Policy, 5(2), 25-36. Available at http://techreports.larc.nasa.gov/ltrs/papers/NASA-95-ir-p25/NASA-95-ir-p25.html.

Nelson, M. L., Maly, K., Shen, S. N. T., & Zubair, M. (1998). NCSTRL+: adding multi-discipline and multi-genre support to the Dienst protocol using clusters and buckets. Proceedings of the IEEE forum on research and technology advances in digital libraries (pp. 128-136), Santa Barbara, CA. Available at http://techreports.larc.nasa.gov/ltrs/PDF/1998/mtg/NASA-98-ieeedl-mln.pdf.

Nelson, M. L. (1999). A digital library for the National Advisory Committee for Aeronautics. NASA/TM-1999-209127. Available at http://techreports.larc.nasa.gov/ltrs/PDF/1999/tm/NASA-99-tm209127.pdf.

Nelson, M. L. (2000). Buckets: smart objects for digital libraries," Ph.D. dissertation, Department of Computer Science, Old Dominion University. Available at http://home.larc.nasa.gov/~mln/phd/.

Paskin, N. (1999). DOI: current status and outlook. D-Lib Magazine, 5(5). Available at http://www.dlib.org/dlib/may99/05paskin.html.

Payette, S. & Lagoze, C. (2000). Policy-Carrying, Policy-Enforcing Digital Objects. In J. Borbinha & T. Baker (eds.), Research and advanced technology for digital libraries, fourth European conference, ECDL 2000, (pp. 144-157), Berlin: Springer.

Phillips, M. S. & Stewart, S. H. (1993). Scientific and technical information of the Langley Research Center for calendar year 1992. NASA Technical Memorandum NASA TM 107706.

Phillips, M. S. & Stewart, S. H. (1995). Scientific and technical information of the Langley Research Center for calendar year 1994. NASA Technical Memorandum NASA TM 109170, Volume I. Available at http://techreports.larc.nasa.gov/ltrs/PDF/tm109170.pdf

Phelps, T. A. & Wilensky, R. (2000). Multivalent documents. Communications of the ACM, 43(6), 83-90.

Pinelli, T. E. (1990). Introduction to National Aeronautics and Space Adminsrtration's scientific and technical information program. Government Information Quarterly 7(2), 123-126.

Rivest, R. (1992). The MD5 message-digest algorithm. Internet RFC-1321. Available at ftp://ftp.isi.edu/in-notes/rfc1321.txt.

Rocha, L. M. (1999). TalkMine and the adaptive recommendation project. Proceedings of the fourth ACM conference on digital libraries (pp. 242-243), Berkeley, CA. Available at http://www.c3.lanl.gov/~rocha/dl99.html.

Roper, D. G., McCaskill, M. K., Holland, S. D., Walsh, J. L., Nelson, M. L., Adkins, S. L., Ambur, M. Y., & Campbell, B. A. (1994). A strategy for electronic dissemination of NASA Langley publications. NASA TM-109172. Available at http://techreports.larc.nasa.gov/ltrs/PDF/tm109172.pdf.

Rothenberg, J. (1995). Ensuring the longevity of digital documents. Scientific American, 272(1), 42-47.

Shafer, K., Weibel, S., Jul, E. & Fausey, J. (1996). Introduction to persistent uniform resource locators. Proceedings of INET 96, Montreal, Canada. Available at http://purl.oclc.org/OCLC/PURL/INET96.

Steiner, J. G., Neuman, C. & Schiller, J. I. (1988). Kerberos: an authentication service for open network systems. Proceedings of the winter 1988 USENIX conference (pp. 191-202), Dallas, TX.

Stewart, S. H. & Phillips, M. S. (1992). Scientific and technical information of the Langley Research Center for calendar year 1991. NASA Technical Memorandum NASA TM 104185.

Stewart, S. H. & Phillips, M. S. (1994). Scientific and technical information of the Langley Research Center for calendar year 1993. NASA Technical Memorandum NASA TM 109050 (1994).

Stewart, S. H. & Phillips, M. S. (1996). Scientific and technical information of the Langley Research Center for calendar year 1995. NASA Technical Memorandum NASA TM 110220, Volume I.

Stewart, S. H. & Phillips, M. S. (1997). Scientific and technical information of the Langley Research Center for calendar year 1996. NASA Technical Memorandum NASA TM 110305, Volume I. Available at http://techreports.larc.nasa.gov/ltrs/PDF/1997/tm/NASA-97-tm110305v1.pdf

Stewart, S. H. & Machie, H. B. (1998). Scientific and technical information of the Langley Research Center for calendar year 1997. NASA Technical Memorandum NASA/TM-1998-206936. Available at http://techreports.larc.nasa.gov/ltrs/PDF/1998/tm/NASA-98-tm206936.pdf

Stewart, S. H. & Machie, H. B. (2000). Scientific and technical information of the Langley Research Center for calendar year 1999. NASA Technical Memorandum NASA/TM-2000-209852. Available at http://techreports.larc.nasa.gov/ltrs/PDF/2000/tm/NASA-2000-tm209852.pdf

Sun, S. X. & Lannom, L. (2001). Handle system overview. Internet Draft. Available at http://www.ietf.org/internet-drafts/draft-sun-handle-system-06.txt.

Task Force on Archiving of Digital Information (1996). Preserving digital information. Available at http://www.rlg.org/ArchTF/.

United States General Accounting Office (1990). NASA is not properly safeguarding valuable data from past missions, GAO/IMTEC-90-1.

Van de Sompel, H., Krichel, T., Nelson, M. L., Hochstenbach, P., Lyapunov, V. M., Maly, K., Zubair, M., Kholief, M., Liu, X. & O' Connell, H. (2000). The UPS prototype: an experimental end-user service across e-print archives. D-Lib Magazine, 6(2). Available at http://www.dlib.org/dlib/february00/vandesompel-ups/02vandesompel-ups.html.

Van de Sompel, H. & Lagoze, C. (2000). The Santa Fe Convention of the Open Archives Initiative. D-Lib Magazine, 6(2). Available at http://www.dlib.org/dlib/february00/vandesompel-oai/02vandesompel-oai.html.

Vinoski, S. (1997). CORBA: integrating diverse applications within distributed heterogeneous environments. IEEE Communications Magazine, 4(2), 46-55.

Wall, L., Christiansen, T. & Schwartz, R. L. (1996). Programming Perl. Sebastopol, CA: O'Reilly & Associates, Inc.

Waugh, A., Wilkinson, R., Hills, B., & Dellóro, J. (2000). Preserving digital information forever. Proceedings of the fifth ACM conference on digital libraries (pp. 175-184), San Antonio, TX.

Walters, J. S. & Schockmel, R. (1998). Applied science publishing in the U.S. government: Failure of congressional policy. Journal of Government Information 25, 95-116.

Weibel, S., Kunze, J., Lagoze, C. & Wolfe, M. (1998). Dublin Core metadata for resource discovery. Internet RFC-2413. Available at ftp://ftp.isi.edu/in-notes/rfc2413.txt.

TABLE 1. Reserved packages in buckets.

Package	Elements Within the Package
_http.pkg	cgi-lib.pl – Steven Brenner's CGI library encoding.e – a list of MIME encoding types mime.e – a list of MIME types
_log.pkg	access.log – messages received by the bucket
_md.pkg	[handle].bib – a RFC-1807 bibliographic file other metadata formats can be stored here, but the .bib file is canonical
_methods.pkg	1 file per public method
_state.pkg	1 file per stored state variable
_tc.pkg	1 file per .tc (terms and condition) file password file .htaccess file

TABLE 2. Bucket API.

Method	Description
add_element	Adds an element to a package
add_method	Adds a method to the bucket
add_package	Adds a package to the bucket
add_principal	Adds a user id to the bucket
add_tc	Adds a T&C file to the bucket
delete_bucket	Deletes the entire bucket
delete_element	Deletes an element from a package
delete_log	Deletes a log file from the bucket
delete_method	Deletes a method from the bucket
delete_package	Deletes a package from the bucket
delete_principal	Deletes a user id from the bucket
delete_tc	Deletes a T&C file from the bucket
display	Displays and disseminates bucket contents
get_log	Retrieves a log file from the bucket
get_preference	Retrieves a preference(s) from the bucket
get_state	Retrieves a state(s) from the bucket
id	Displays the bucket's unique id
lint	Checks the buckets internal consistency
list_logs	Lists all the log files in the bucket
list_methods	Lists all the methods in the bucket
list_principals	Lists all the user ids in the bucket
list_source	List the method source
list_tc	Lists all the T&C files in the bucket
metadata	Displays the metadata for the bucket
pack	Returns a "bucket-stream"
set_metadata	Uploads a metadata file to the bucket
set_preference	Changes a bucket preference
set_state	Changes a bucket state variable
set_version	Changes the version of the bucket
unpack	Overlays a "bucket-stream" into the bucket
version	Displays the version of the bucket

TABLE 3. OAI verbs.

Verb	Function
Identify	machine readable description of archive
ListMetadataFormats	metadata formats supported by archive
ListSets	sets defined by archive
ListIdentifiers	OAI unique ids contained in archive
ListRecords	listing of all records
GetRecord	listing of a single record

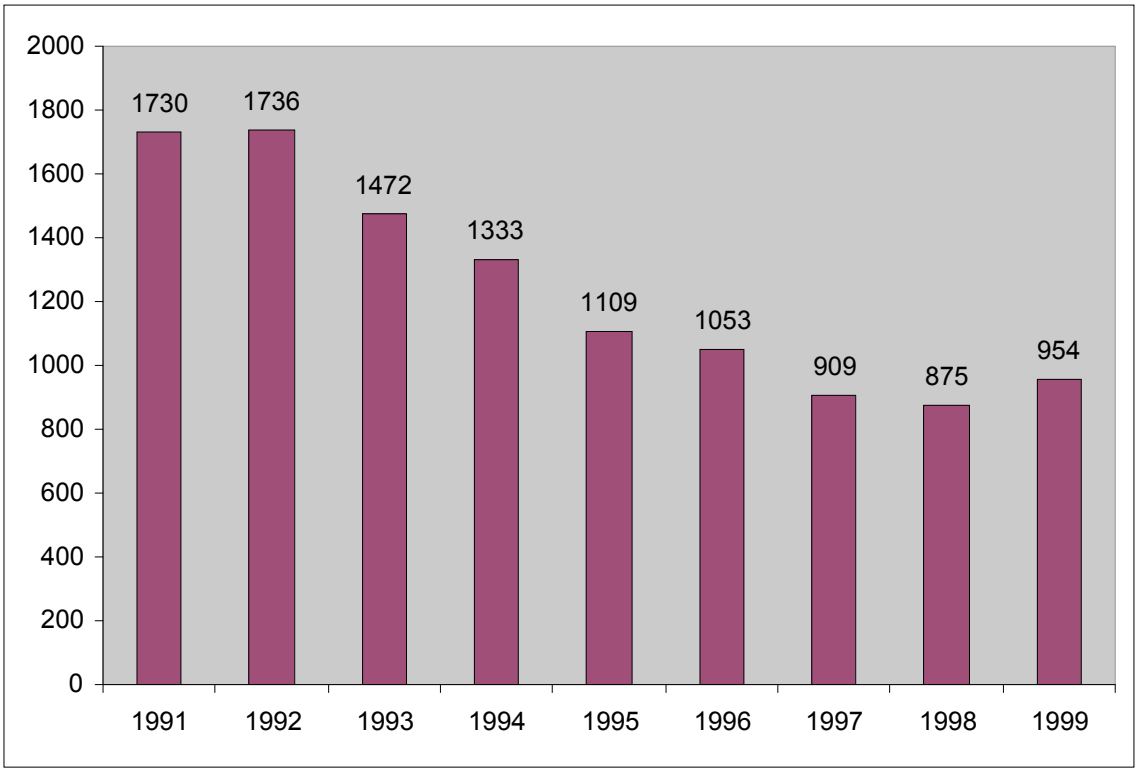

Figure 1. Production of NASA Langley Research Center Technical Publications, 1991-1999 (Stewart & Phillips, 1992; Phillips & Stewart, 1993; Stewart & Phillips, 1994; Phillips & Stewart, 1995; Stewart & Phillips, 1996; Stewart & Phillips, 1997; Stewart & Machie, 1998; Machie & Stewart, 1999; Stewart & Machie, 2000)

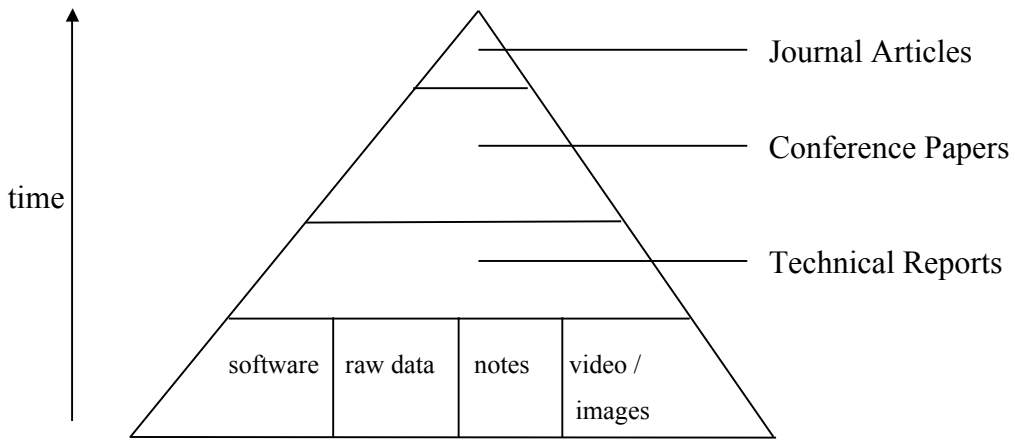

Figure 2. Pyramid of publications rests on unpublished STI.

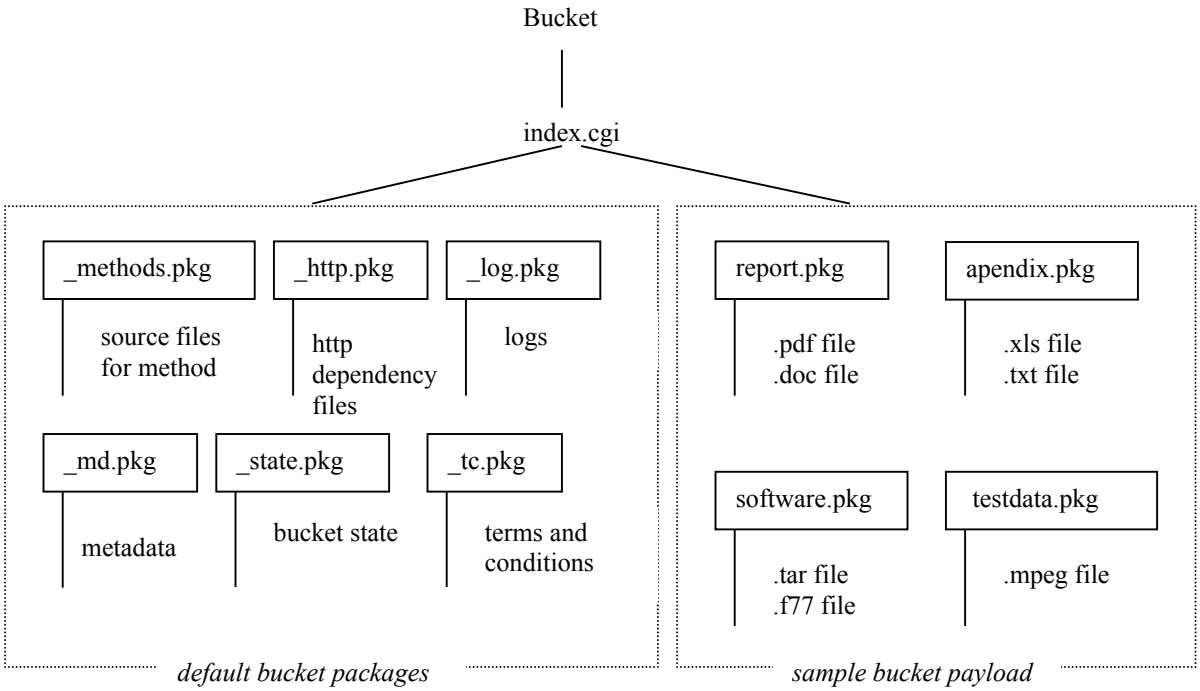

Figure 3. Model Bucket structure.

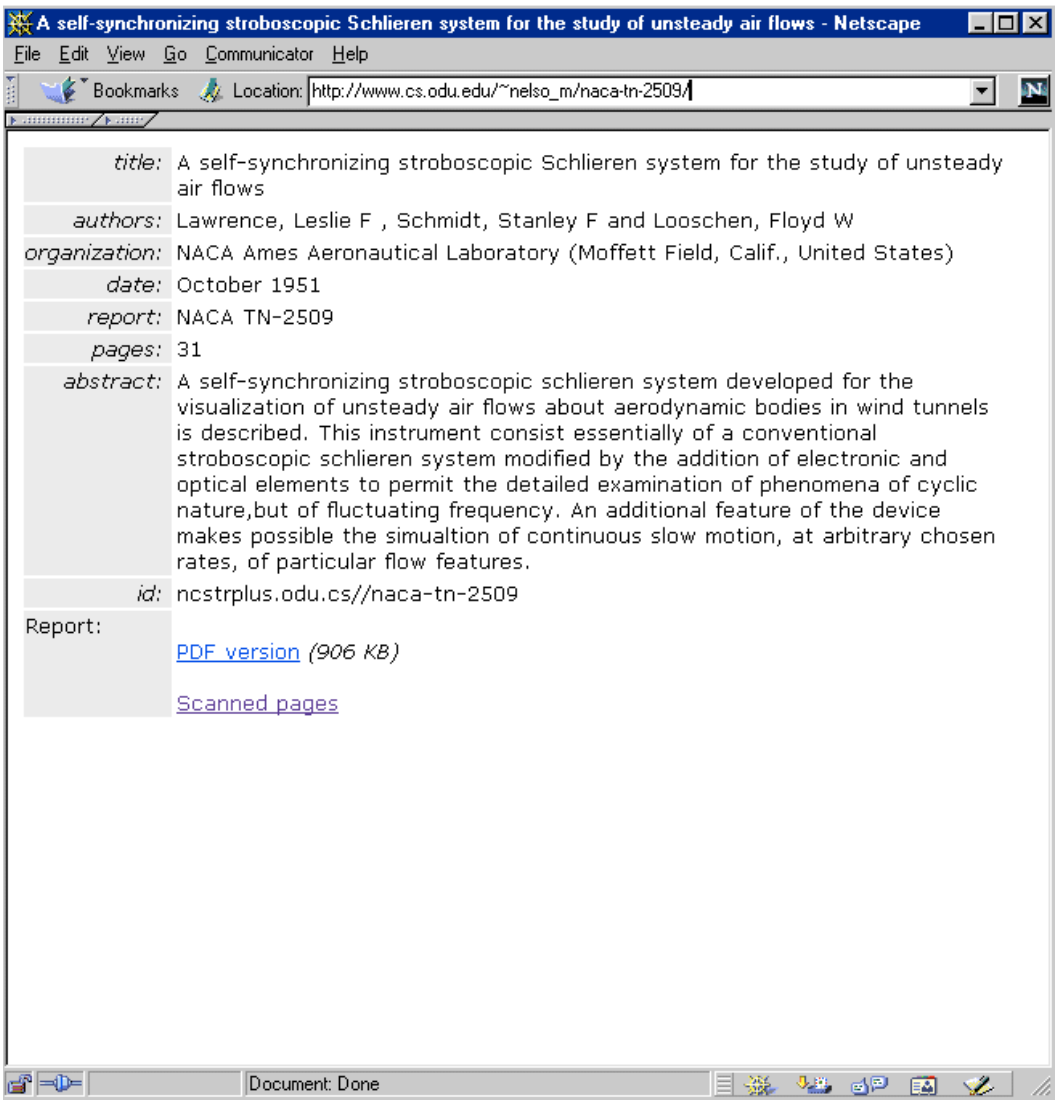

Figure 4. The default display method reveals the bucket contents in a human readable format.

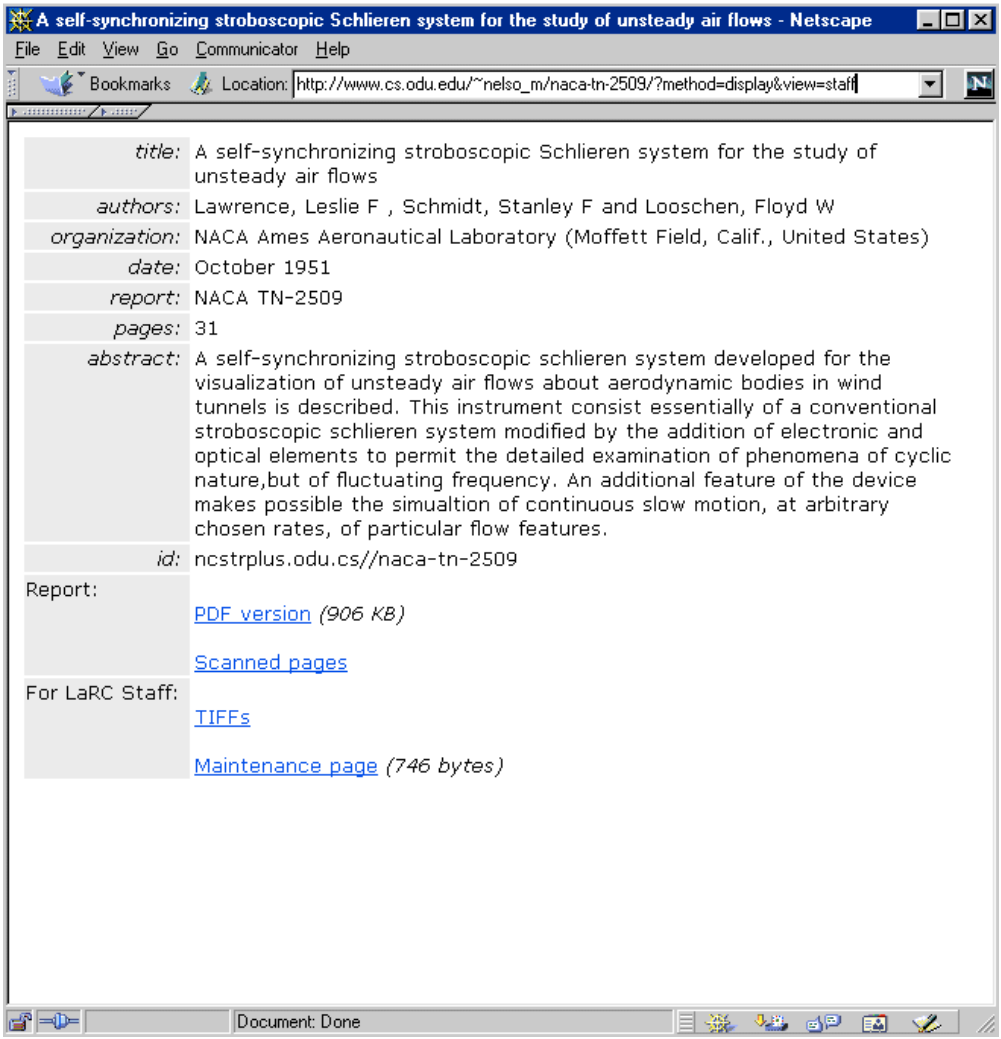

Figure 5. Extra options are available for library staff.

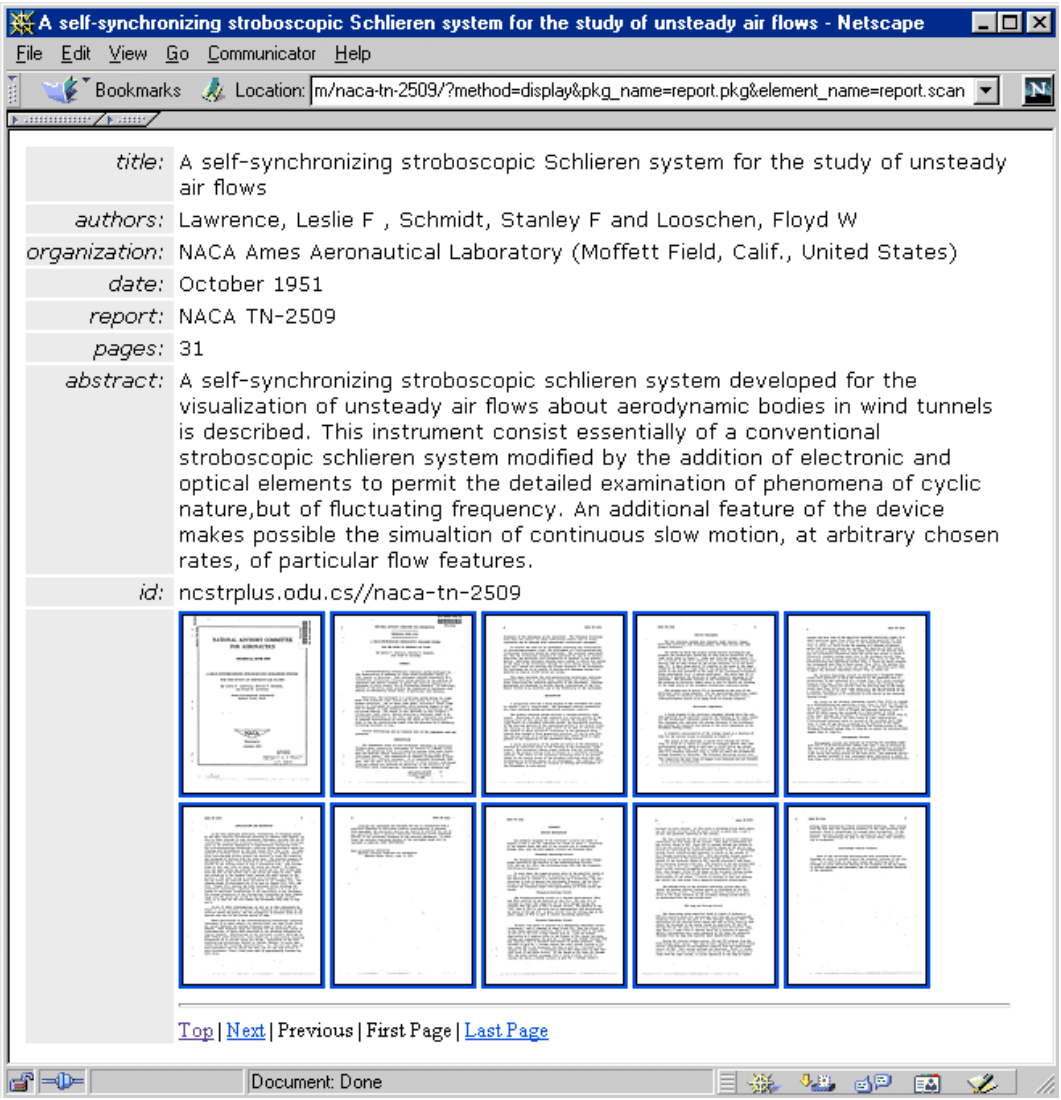

Figure 6. The first 10 scanned thumbnails within the bucket are displayed, along with pagination control.