Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries, Paris, France, September 22-24, 1999, pp. 453-464

SODA: Smart Objects, Dumb Archives

Michael L. Nelson[1], Kurt Maly[2], Mohammad Zubair[2], Stewart N. T. Shen[2]

[1] NASA Langley Research Center, MS 158, Hampton, Virginia, USA 23681
m.l.nelson@larc.nasa.gov
[2] Old Dominion University, Computer Science Department, Norfolk, Virginia, USA 23592
{maly, zubair, shen}@cs.odu.edu

Abstract. We present the Smart Object, Dumb Archive (SODA) model for digital libraries (DLs). The SODA model transfers functionality traditionally associated with archives to the archived objects themselves. We are exploiting this shift of responsibility to facilitate other DL goals, such as interoperability, object intelligence and mobility, and heterogeneity. Objects in a SODA DL negotiate presentation of content and handle their own terms and conditions. In this paper we present implementations of our smart objects, buckets, and our dumb archive (DA). We discuss the status of buckets and DA and how they are used in a variety of DL projects.

1 Introduction

The Smart Object, Dumb Archive (SODA) model for digital libraries (DLs) was developed within the context of NCSTRL+ [15], the joint NASA Langley Research Center and Old Dominion University extension of the Networked Computer Science Technical Report Library (NCSTRL) [1]. The premise of the SODA model is a separation of responsibilities: associating with digital libraries such traditional value-added services as indexing and searching; with digital objects their individual properties as distinguished from those of a collection; and with archives guaranteed access over a period of time. To this end we have developed buckets, data objects tailored for DL use, that enforce their own terms and conditions, negotiation and presentation of content, and maintain their own metadata. Many of the functions traditionally associated with archives have been "pushed down" into the buckets themselves, resulting in "smarter" objects and "dumber" archives.

Buckets are thus a special class of digital objects that are aggregative and intelligent agents. Buckets are DL-protocol independent, and due to their self-contained nature, can exist outside of DLs altogether. Buckets provide the mechanism (not the policy) for grouping related information objects into a single logical entity for archiving. We are designing buckets to contain intelligent agents so they can communicate with each other, people, and arbitrary network services as well as perform computational and self-modifying tasks. Buckets are described further in [14].

Archives in the SODA model correspond to the lowest level - core archives - of the Kahn/Wilensky Framework (KWF) [2]. In that spirit we believe there should be only very limited functionality associated with an archive. Specifically, the functions an archive should support are those of: add, delete, retrieve, list all objects and set/get metadata about the archive.

1.1 Terminology

Since there is no consensus on DL terminology, we use the following definitions for this discussion:

_ *digital library services*- the “user” functionality and interface: searching, browsing, usage analysis, citation analysis, selective dissemination of information (SDI), etc.
_ *archive* - managed sets of data objects. DLs can poll archives to learn of newly published data objects, for example.
_ *data object* - the stored and trafficked digital content. These can be simple files (e.g., PDF or PS files), more sophisticated objects such as buckets.

Figure 1 illustrates that hierarchical nature of DLs, archives, and buckets. A DLS is shown as a single entity, but this logical entity could be a distributed set servers. Note that although users can communicate with archives, we envision that archives will largely function only as middleware – enabling a DLS to locate buckets. Users will find buckets through a DL interface, and once found they will interact with the buckets themselves.

Other DL models are possible (Table 1). The Smart Objects, Smart Archives (SOSA) model is possible, even likely to be the "default" DL of the future. However, to highlight the functionalities of buckets, we introduce them in the SODA context. Note that the Dumb Object, Smart Archive (DOSA) model describes the state of most current DLs, and the Dumb Object, Dumb Archive (DODA) model is an accurate description of early DLs.

Table 1. Archive Design Space

	Smart Archives	Dumb Archives
Smart Objects	SOSA DL Example: none known	SODA DL Example: NCSTRL+
Dumb Objects	DOSA DL Example: NCSTRL	DODA DL Example: an anonymous FTP server with .ps.Z files

1.2 Motivation

We have been involved with a number of high traffic production NASA DLs since 1994, including the Langley Technical Report Server [17], the NASA Technical

Report Server [16], and the NACA Report Server [12]. One thing we have observed from http log files is a surprising number of people do not find the NASA and NACA publications via the NASA and NACA DLs. Since the full contents of the NASA DLs are browsable, both the abstract lists and the reports are indexed by web crawlers, spiders and the like. Users are formulating complex queries to services such as Yahoo, Altavista, Lycos, Infoseek, etc. We presume this is indicative of the resource discovery problem: people start there because they do not know all the various DLs themselves; and the meta-searching problem: they are trusting these services to search many sources, not just the holdings of a single DL.

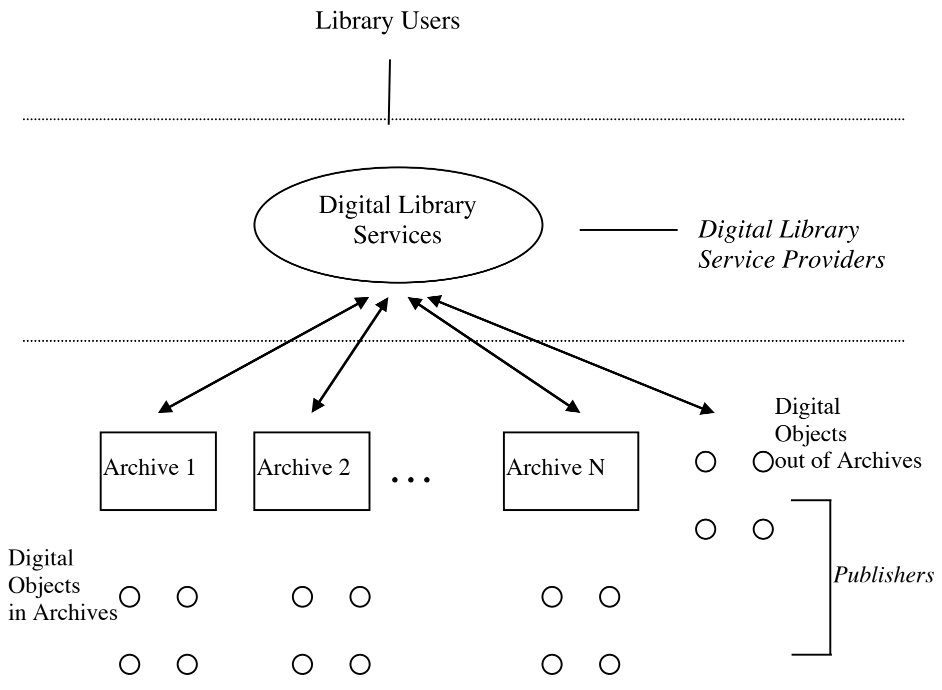
Library Users
Digital Library Services
Digital Library Service Providers
Archive 1
Archive 2
. . .
Archive N
Digital Objects out of Archives
Digital Objects in Archives
Publishers

Fig. 1. Access in the DL Hierarchy

Although we believe we have built attractive and useful interfaces for the NASA DLs, our main concern is that people have access to NASA content and not that they use a specific DL interface. It is desirable that NASA publications are indexed by many services. Since there are several paths to the information object, the information object must be a first class network citizen, handling presentation, terms and conditions, and not depending on archive functionality. Buckets implement the object as a first class citizen idea, and both buckets and the DA software facilitate greater dissemination of the material by making it easier for the holdings to be found and indexed by third party services.

1.3 Background

The NCSTRL+ project is based on the creation of buckets and the extension of the Dienst [8] protocol. Dienst is a collection of DL services that receive messages encoded and transmitted via hypertext transfer protocol (http). Objects in Dienst are stored in directories, and are accessed through the Repository service. Metadata for the objects are stored in RFC-1807 format [9]. In addition to changing Dienst to properly handle buckets, we have added a new verb, Recluster, to the User Interface Service to assist in dynamically changing the display of search results.

2 Buckets: Smart Objects

Buckets are object-oriented container constructs in which logically grouped items can be collected, stored, and transported as a single unit. 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 or degrade over time. Some portions such as software, can go into separate archives (i.e., COSMIC or the Langley Software Server) but this leaves the researcher to re-integrate the information tuple by selecting pieces from multiple archives. Most often, the software and other items, such as datasets are simply discarded. After 10 years, the manuscript is almost surely the only surviving artifact of the information tuple.

Large archives could have buckets with many different functionalities. Not all bucket types or applications are known at this time. However, we can describe a generalized bucket as containing many formats of the same data item (PS, Word, Framemaker, etc.) but more importantly, it can also contain collections of related non-traditional STI materials (manuscripts, software, datasets, etc.) Thus, buckets allow the digital library to address the long standing problem of ignoring software and other supportive material in favor of archiving only the manuscript [21] by providing a common mechanism to keep related STI products together. The current semantics of buckets include a two-level structure: "elements", which are the unit of storage in buckets, and "packages", which are groups of elements. Figure 2 illustrates a typical bucket in a NASA DL application.

Our bucket prototypes are written in Perl 5, and make use of the fact that Dienst uses http as a transport protocol. Like Dienst, bucket metadata is stored in RFC-1807 format, and package and element information is stored in newly defined optional and repeatable fields. 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. However, this is implementation specific. The bucket API defines all operations on buckets. The bucket is accessible through a Common Gateway Interface (CGI) script that parses the messages and enforces terms and conditions, and negotiates presentation to the WWW client. The bucket presentation format is currently encoded within the bucket, but we are currently planning to model presentation requirements using the Resource Description Framework (RDF) [10] to provide a mechanism for providing dynamic presentation templates that can exploit known semantics during presentation.

The philosophy of Dienst is to minimize the dependency on HTTP. Except for the User Interface service, Dienst does not make specific assumptions about the existence of HTTP or the Hypertext Markup Language (HTML). However, Dienst does make very explicit assumptions about what constitutes a document and its related data formats. Built into the protocol are the definitions of PostScript, ASCII text, inline images, scanned images, etc. We feel that tightly coupling the DL protocol with knowledge of individual file formats reduces the flexibility of the DL protocol, making it less adaptable to new or locally defined data types and data relations.

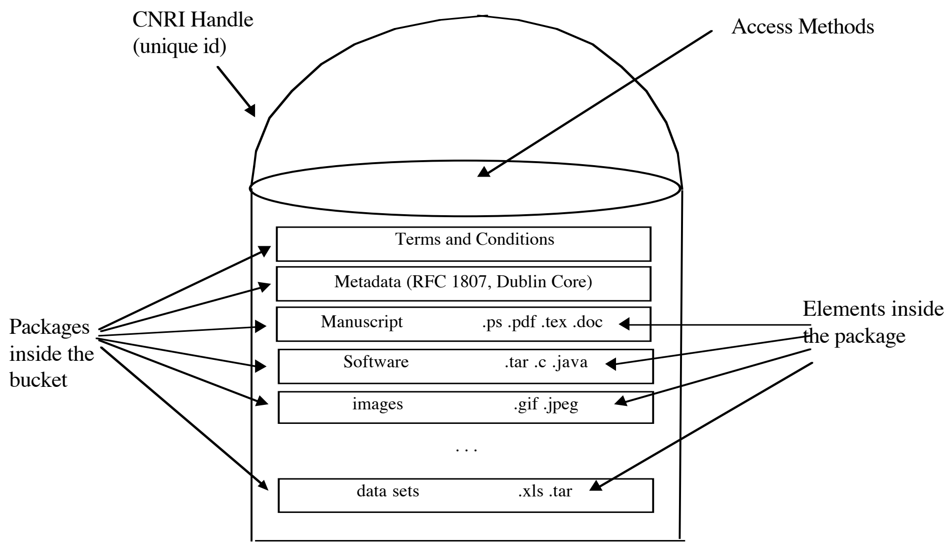

Fig. 2. A Typical NASA DL Bucket

We favor making Dienst less knowledgeable about dynamic topics such as file format, and making such knowledge the responsibility of buckets. In NCSTRL+, Dienst is used as an index, search, and retrieval protocol. When the user selects an entry from the search results, Dienst would normally have the local User Interface service use the Describe verb to peer into the contents of the documents directory (including the metadata file), and Dienst itself would control how the contents are presented to the user. In NCSTRL+, the final step of examining the directory structure is skipped, and NCSTRL+ issues a URL redirect to the bucket. At this point, the user is communicating with the bucket and not Dienst or any other archive. The default method for an index.cgi is the display method, so the user notices little difference in operation between NCSTRL and NCSTRL+.

The full list of bucket methods are discussed in [14]. Table 2 list some common methods defined on a particular test bucket. Embedding this archive-like functionality in the buckets comes at the expense of additional storage overhead, approximately 80KB per bucket. However, we consider this trivial in comparison to the size of typical NASA buckets (often several MBs), and with respect to the additional functionality of object intelligencce, object heterogeneity, DL protocol independence, and object mobility.

Table 2. Some Sample Bucket Methods

Methods and Arguments	Description
http://dlib.cs.odu.edu/bucket/?method=display *or* http://dlib.cs.odu.edu/bucket/	Displays the bucket's contents in HTML. The default method.
http://dlib.cs.odu.edu/bucket/?method=list_principals (this bucket's appendices are restricted to "maly" / "maly")	Lists the principals (entries in the password file). Access can be restricted to specific principals.
http://dlib.cs.odu.edu/bucket/?method=list_methods	Lists all the methods known by this bucket.
http://dlib.cs.odu.edu/bucket/?method=list_source&target=display	Lists the source code for the "display" method.
http://dlib.cs.odu.edu/bucket/?method=list_tc&target=display.tc	Lists the terms and conditions for the "display" method.
http://dlib.cs.odu.edu/bucket/?method=list_logs	Lists the names of all logs kept by the bucket.
http://dlib.cs.odu.edu/bucket/?method=get_log&log=access.log	Displays the access log.
http://dlib.cs.odu.edu/bucket/?method=id	Displays the bucket's handle.
http://dlib.cs.odu.edu/bucket/?method=metadata	Returns the bucket metadata in RFC-1807 format.

3 DA: Dumb Archives

The use of buckets or other smart objects does not necessitate the use of dumb archives; it is possible to use buckets in a number of DL and WWW systems. However, we are implementing DA (dumb archive) as a reference implementation demonstrating the low level of functionality required for use in the SODA model. Table 3 lists the basic methods defined for DA.

The DA is essentially a set manager - notice the DA has no search capabilities. The DA'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. And 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, we expect the NASA digital publishing model to begin with technical publications, after passing through their respective internal quality control, to be placed in a NASA archive. The NASA DL would poll this archive to learn the location buckets published within that 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 possibility novel and unique methods for searching or browsing. Or perhaps the DL collects all the metadata, then performs additional filtering to 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 we view all digitally available publications 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. Figure 3 illustrates this publishing model.

Table 3. Methods for a Dumb Archive

Method	Description
put	insert a data object into the archive
delete	remove a data object from the archive
list	display the holdings of the archive
info	display metadata about the archive
get	redirects to the object’s URL or URN

4 Discussion

Although buckets and SODA were initially implemented using modified versions of Dienst, it should be stressed that neither buckets nor SODA require Dienst to operate. Indeed, a bucket design goal is to provide sophisticated digital objects for DLs that do not require Dienst, or any other specific DL protocol, to be used. In our internal applications, we regularly use buckets without Dienst. To applications that know to exploit them, buckets offer much functionality. To applications that are not bucket aware, buckets appear as regular HTML pages. For example, it would be easy to build a DL using a webcrawler search engine (such as Excite for Web Servers or Ultraseek Server). This would not be easily accomplished for data existing only within a Dienst archive.

There may be situations in which buckets are unnecessary. For large homogeneous collections, the storage overhead and additional administrative work of managing both buckets and archives may be undesirable. For a DL that may never be more than a DOSA or DODA DL, buckets are probably unnecessary. However, buckets are motivated from of our own production DL experiences in which latent or creeping requirements demanded that dumb objects eventually become smart.

Similarly, the motivation for SODA comes from our negative experiences in transitioning from one DL system to the next, and have the same body of content indexed by multiple systems. We believe that SODA builds the foundation for object mobility, object-level heterogeneity and DL protocol level heterogeneity. We intend to test these goals when we transition buckets and SODA from our prototypes to production NASA DLs.

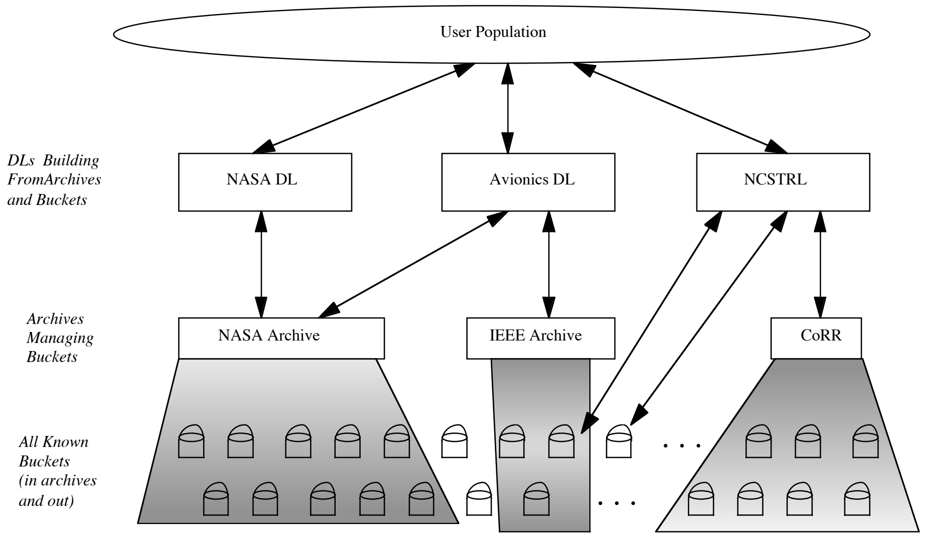
User Population
DLs Building FromArchives and Buckets
NASA DL
Avionics DL
NCSTRL
Archives Managing Buckets
NASA Archive
IEEE Archive
CoRR
All Known Buckets (in archives and out)
. . .
. . .

Fig. 3. The SODA Publishing Model

5 Status

The NCSTRL+ DL interface is based on our extensions to the Dienst protocol to provide a testbed for experimentation with buckets, clusters, and interoperability. "Clustering" is an advanced searching and browsing capability that allows dynamic clustering of holdings based on subject, institution, archival type and terms and conditions. The NCSTRL+ interface can be accessed at: http://dlib.cs.odu.edu/ncstrlplustool.html

Our long-term plans call for the conversion of the NASA DLs to buckets and NCSTRL+. At this point, the NCSTRL+ project is converting the over 1800 items in the Langley Technical Report Server (LTRS) to buckets. The lessons learned in the LTRS conversion is described in [13]. The buckets created in that conversion process were stored in a dumb, stand-alone archive which was then indexed into NCSTRL+. Therefore, having the handle for a bucket allows a user to retrieve the bucket from the archive; on the other hand the user can search NCSTRL+ to find a bucket. In either case the bucket itself will handle the presentation.

Additionally, we have developed a set of tools to aid in the creation, tracking and management of buckets and enforcing publishing and maintenance policies for archives. The tools have been used to create a testbed for NCSTRL+ which, at this time, runs on three NCSTRL+ servers with index service for five archives. Since NCSTRL+ can access other Dienst collections we can extend searches to all of NCSTRL, CoRR, and D-Lib Magazine as well.

Other active bucket development areas include: the creation of "light-weight buckets" that provide a author specified subset of functionality to save on storage overhead and the creation of XML specified bucket ontologies. These will shield users from the two-level constructs of packages and elements and allow the storage and interaction of arbitrarily complex hierarchies that represent real world objects (i.e., "assignments" within a "university class" bucket).

6 Related Work

There is extensive research in the area of redefining the concept of "document" or providing container constructs. In this section we examine some of these projects and technologies that are similar to buckets, as well as projects that similar capabilities as DA. Although buckets as intelligent agents is not described in this paper, we also note that we are unaware of other attempts to make archival entities intelligent.

6.1 Bucket-Like Projects

Buckets are most similar to the digital objects first described in the Kahn/Wilensky Framework [2], and its derivatives such as the Warwick Framework containers [6] and the more recent Flexible and Extensible Digital Object Repository Architecture (FEDORA) [18]. In FEDORA, DigitalObjects are containers, which aggregate one or more DataStreams. DataStreams are accessed through an Interface, and an Interface may in turn be protected by an Enforcer. The significant design difference between the KWF derivatives and buckets is that KWF DigitalObjects are tightly tied to the archive and protocol that hold and serves them.

Multivalent documents [19] appear similar to buckets at first glance. However, the focus of multivalent documents is more on expressing and managing the relationships of differing “semantic layers” of a document, including language translations, derived metadata, annotations, etc. There is not an explicit focus on the aggregation of several existing data types into a single container.

E-commerce applications are producing a number of bucket-like projects. One example is IBM's cryptolopes [4], which are designed to allow for unlimited distribution of that data objects, but controlled access to their contents. Similarly, DigiBox [20] has been developed with the goal “to permit proprietors of digital information to have the same type and degree of control present in the paper world” [20]. As such, the focus of the DigiBox capabilities are heavily oriented toward cryptographic integrity of the contents, and not so much on the less stringent demands of the current average digital library. There appears to be no hooks to make either DigiBoxes or Cryptolopes intelligent agents. DigiBox and Cryptolope are commercial endeavors and are thus less suitable for our research purposes.

To a lesser extent, buckets are not unlike some of the proposals from various experimental filesystems and scientific data types. The Extensible File System (ELFS) [3] provides an abstract notion of “file” that includes both aggregation, data format heterogeneity, and high performance capabilities (striping, pre-fetching, etc.). While ELFS is designed primarily for a non-DL application (i.e., high-performance computing), it is typical of an object-oriented approach to file systems, with generic access APIs hiding the implementation details from the programmer.

The Hierarchical Data Format (HDF) and related formats (netCDF, HDF-EOS, etc.) is a multi-object, aggregative data format that is alternatively: raw file storage, the low-level I/O routines to access the raw files, an API for higher level tools to access, and a suite of tools to manipulate and analyze the files [11 22]. While HDF is mature and has an established user base, it is largely created by and for the earth and atmospheric sciences community, and this community's constraints limits the usefulness of HDF as a generalized DL application. It is worth noting, however, that buckets of HDF files are entirely possible and appropriate.

6.2 DA-Like Projects

DA is interesting because of what it leaves out, not what implements. As the name implies, there are any number of more sophisticated archive related projects and technologies. For example, the proposed Repository Access Protcol (RAP) [7] reveals many the same operations of DA (VERIFY, DEPOSIT, DELETE, etc.), but it defines separate explicit ACCESS operations for both the digital object and its metadata. Such concepts in SODA have been removed from the DA and placed within the bucket itself.

The Dienst protocol has some DA-like concepts as well. In particular, the Repository Service in Dienst implements a List-Contents verb, the LibMgt Service implements a Submit verb, etc. However, the main function of the Repository Service in Dienst is to regulate access to the items in the repository, through verbs such as Body and Page. Again, in SODA these functions are pushed down into the buckets.

The Dienst research group have proposed a more recent service, the Collection Service [5]. This service is more like DA than the previous examples, in that its purpose is to group together arbitrary network objects based on some criteria. However, future plans for the collection service call for it to be involved in operations such as query routing, which are obviously beyond the scope of the DA. When the Collection Service is available for testing, it may be a good candidate to implement a SOSA model DL.

7 Conclusions

The SODA DL model was created for the NCSTRL+ project to facilitate DL interoperability and to increase the scope and nature of the availability of archived data objects. SODA shifts many functions associated with archives to the archived objects themselves. We have developed aggregative and intelligent archival entities, buckets, to handle this shift in responsibility. Buckets can exist in a number of DL archives, or outside archives altogether and responsible for presentation of the their contents and enforcing their own terms and conditions. To services that are not bucket-aware, buckets appear as ordinary HTML pages. The dumb archive we have developed, DA, provides just enough functionality to illustrate the role of an archive as the middle layer in the SODA DL hierarchy. SODA facilitates DL interoperability by clearly separating the roles of a DL, an archive, and the object itself. Finally, the SODA model facilitates wider dissemination of holdings by making it easy for third party services to find and index buckets.

References

1. J. Davis & C. Lagoze: The Networked Computer Science Technical Report Library, Cornell CS TR96-1595, July, 1996. http://cs-tr.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.cornell/TR96-1595
2. R. Kahn & R. Wilensky: A Framework for Distributed Digital Object Services, cnri.dlib/tn95-01, May 1995. http://www.cnri.reston.va.us/home/cstr/arch/k-w.html
3. J. F. Karpovich, A. S. Grimshaw, & J. C. French: Extensible File Systems (ELFS): An Object-Oritented Approach to High Performance File I/O, Proceedings of the Ninth Annual Conference on Object-Oriented Programming Systems, Languages and Applications, October 1994, pp. 191-204.
4. U. Kohl, J. Lotspiech, M. A. Kaplan: Safeguarding Digital Library Contents and Users: Protecting Documents Rather Than Channels, D-Lib Magazine, September 1997. http://www.dlib.org/dlib/september97/ibm/09lotspiech.html

5. C. Lagoze & D. Fielding: Defining Collections in Distributed Digital Libraries. D-Lib Magazine, November 1998. http://www.dlib.org/dlib/november98/lagoze/11lagoze.html
6. C. Lagoze, C. A. Lynch & R. Daniel: The Warwick Framework: A Container Architecture for Aggregating Sets of Metadata, Cornell Computer Science Technical Report TR96-1593, July 1996. http://cs-tr.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.cornell/TR96-1593
7. C. Lagoze, R. McGrath, E. Overly & N. Yeager: A Design for Inter-Operable Secure Object Stores (ISOS), Cornell CS TR95-1558, November 1995. http://cs-tr.cs.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell/TR95-1558
8. C. Lagoze, E. Shaw, J. R. Davis, & D. B. Krafft: Dienst: Implementation Reference Manual, Cornell University Computer Science TR95-1514, May 1995. http://cs-tr.cs.cornell.edu/Diesnt/UI/2.0/Describe/ncstrl.cornell/TR95-1514
9. R. Lasher & D. Cohen: A Format for Bibliographic Records, Internet RFC-1807, June 1995. http://info.internet.isi.edu/in-notes/rfc/files/rfc1807.txt
10. E. Miller: An Introduction to the Resource Description Framework, D-Lib Magazine, May 1998. http://www.dlib.org/dlib/may98/miller/05miller.html
11. NCSA HDF Home Page, http://hdf.ncsa.uiuc.edu
12. M. Nelson: A Digital Library for the National Advisory Committee for Aeronautics, NASA TM-1999-209127, April 1999. http://techreports.larc.nasa.gov/ltrs/PDF/1999/tm/NASA-99-tm209127.pdf
13. M. L. Nelson, K. Maly, S. N. T. Shen, & M. Zubair: Generalizing an Existing Digital Libraries, Old Dominion University Computer Science TR-99-01, February 1999. http://cs-tr.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.odu_cs//TR_99_01
14. M. L. Nelson, K. Maly, S. N. T. Shen, & M. Zubair: Buckets: Aggregative, Intelligent Agents for Publishing, Webnet Journal, 1(1), 1999, pp. 58-66. (Also available as NASA TM-1998-208419; http://techreports.larc.nasa.gov/ltrs/PDF/1998/tm/NASA-98-tm208419.pdf)
15. M. L. Nelson, K. Maly, S. N. T. Shen, & M. Zubair: NCSTRL+: Adding Multi-Discipline and Multi-Genre Support to the Dienst Protocol Using Clusters and Buckets, Proceedings of Advances in Digital Libraries 98, Santa Barbara, CA, April 22-24, 1998. http://techreports.larc.nasa.gov/ltrs/PDF/1998/mtg/NASA-98-ieeedl-mln.pdf
16. M. L. Nelson, G. L. Gottlich, D. J. Bianco, S. S. Paulson, R. L. Binkley, Y. D. Kellogg, C. J. Beaumont, R. B. Schmunk, M. J. Kurtz & A. Accomazzi: The NASA Technical Report Server, Internet Research: Electronic Networking Applications and Policy, 5(2), 1995, pp. 25-36.
17. M. L. Nelson , G. L. Gottlich & D. J. Bianco: World Wide Web Implementation of the Langley Technical Report Server, NASA TM-109162, September 1994.
18. S. Payette & C. Lagoze: Flexible and Extensible Digital Object and Repository Architecture, In C. Nikolaou & C. Stephanidis (eds.) Research and Advanced Technology for Digital Libraries, Second European Conference, ECDL 98, Lecture Notes in Computer Science, Vol. 1513, 1998, pp. 41-59.
19. T. A. Phelps & R. Wilensky: Multivalent Documents: Inducing Structure and Behaviors in Online Digital Documents: Proceedings of the 29th Hawaii International Conference on System Sciences, Maui, HI, January 3-6, 1996.
20. O. Sibert, D. Bernstein & D. Van Wie: DigiBox: A Self-Protecting Container for Information Commerce, Proceedings of the 1st USENIX Workshop on Electronic Commerce, New York, NY, July, 1995.
21. J. Sobieszczanski-Sobieski: A Proposal: How to Improve NASA-Developed Computer Programs, NASA CP-10159, 1994, pp. 58-61.
22. I. Stern: Scientific Data Format Information FAQ, 1995. http://www.cv.nrao.edu/fits/traffic/scidataformats/faq.html