D-Lib Magazine
March 1999

Volume 5 Issue 3
ISSN 1082-9873

Smart Objects, Dumb Archives

A User-Centric, Layered Digital Library Framework

Kurt Maly
Old Dominion University
Computer Science Department
Norfolk, VA 23592
maly@cs.odu.edu

Michael L. Nelson
NASA Langley Research Center
MS 158
Hampton, VA 23681
m.l.nelson@larc.nasa.gov

Mohammad Zubair
Old Dominion University
Computer Science Department
Norfolk, VA 23592
zubair@cs.odu.edu

Abstract

Currently, there exist a large number of superb digital libraries, all of which are, unfortunately, vertically integrated and all presenting a monolithic interface to their users. Ideally, a user would want to locate resources from a variety of digital libraries dealing only with one interface. A number of approaches exist to this interoperability issue exist including: defining a universal protocol for all libraries to adhere to; or developing mechanisms to translate between protocols. The approach we illustrate in this paper is to push down the level of universal protocols to one for digital object communication and for communication for simple archives. This approach creates the opportunity for digital library service providers to create digital libraries tailored to the needs of user communities drawing from available archives and individual publishers who adhere to this

standard. We have created a reference implementation based on the hyper text transfer protocol (http) with the protocols being derived from the Dienst protocol. We have created a special class of digital objects called buckets and a number of archives based on a NASA collection and NSF funded projects. Starting from NCSTRL [2] we have developed a set of digital library services called NCSTRL+ and have created digital libraries for researchers, educators and students that can each draw on all the archives and individually created buckets.

Introduction

Currently, there exist a large number of superb digital libraries (DLs), all of which are, unfortunately, vertically integrated and all presenting a monolithic interface to their users. Ideally, a user would want to locate resources from a variety of digital libraries dealing only with one interface. A number of approaches to this interoperability issue exist including: defining a universal protocol for all libraries to adhere to; or developing mechanisms to translate between protocols. The approach we illustrate in this paper is to push down the level of universal protocols to one for digital object communication and for communication for simple archives. This creates the opportunity for digital library service providers to integrate digital libraries tailored to the needs of user communities drawing from available archives and individual publishers who adhere to this standard. It argues for generally available tools to create (publish) rich digital objects and it further argues that organizations exist which will put their imprimatur on these objects as a well understood standard of quality.

For a Digital Library Service Provider (DLSP) to build a successful digital library, there have to be standard methods to interact with archives and digital objects. In this paper we propose such a standard, and we have created a reference implementation based on http. Since we can no longer be sure which digital library will be used for the discovery and presentation of an object, we feel that it is necessary to evolve the notion of the object and to imbue it with greater functionality and responsibility. We argue for self-contained, intelligent, and aggregative DL objects that are capable of enforcing their own terms and conditions, negotiating access, and displaying their contents. We call this specialized class of digital objects "buckets". Once a DLSP has delivered to the customer a location of a bucket it is up to the bucket to interact with the customer. On the other hand, in our model archives are simply collections of buckets characterized by some management policy that controls publishing. It is the archive-owning organization that negotiates with the DLSP for access to the archive’s buckets. Since all the important presentation services are associated with the buckets themselves, the archive services can be few and simple such as "list locations of all buckets".

In our view, a DLSP builds a DL by:

- Identifying a user group;
- Identifying archives holding buckets of interest and individual bucket owners;

- Negotiating terms and conditions with publishing organizations (archive and individual bucket owners);
- Creating indices of appropriate subsets by interacting with buckets for their metadata;
- Creating DL services such as search and browse;
- Creating user interaction services such as authentication and billing.

We have a reference implementation, NCSTRL+ [18], which implements a DL using the Dienst protocol and http services. (Other implementations, such as CORBA or simple TCP/IP, are possible.) The Digital Library Services (DLSs) are provided by using the basic core of Dienst for searching, browsing and similar services. The archive functionality was originally implemented using a modified version of Dienst, but we are now transitioning to a simpler archive system. We have created a special class of digital objects called buckets and a number of archives based on a NASA collection and NSF funded projects.

The rest of the paper is organized as follows. In the next section, we discuss the model supporting DLSP, SODA. Following, we discuss our reference implementation, to date, of the SODA model. We then discuss our status and future plans. This is followed by a brief discussion of related work.

The SODA Model

We present a model that defines DLs as composed of 3 strata (Figure 1):

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

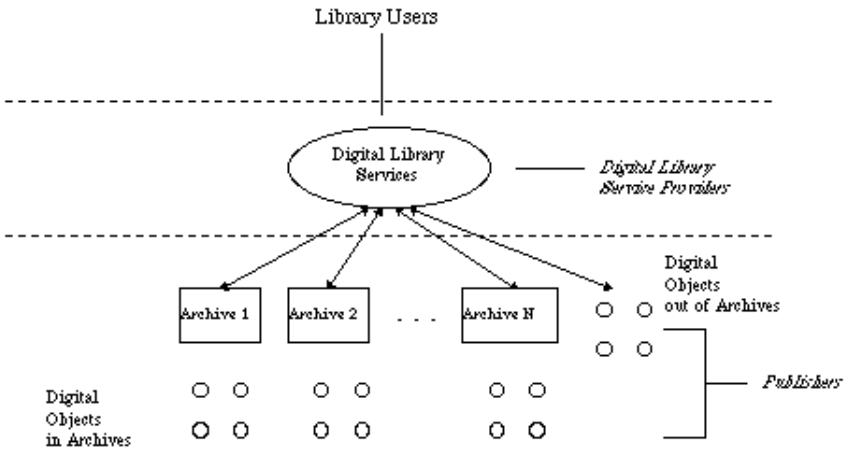

Figure 1: The Three Strata of DLs

In most DLs, the digital library services (DLS) and the archive functionality are tightly coupled. A digital object is placed in an archive, and this placement uniquely determines in which DL it appears. We believe that if there is not a 1-1 mapping between archives and DLs, but rather a N-M mapping, the capacity for interoperability is greatly advanced. A DL can draw from many archives, and likewise, an archive can contribute its contents to many DLs.

However, since we can no longer be sure which DL will be used for the discovery and presentation of an object, we feel it is necessary to evolve the notion of the object and to imbue it with greater functionality and responsibility. We argue for self-contained, intelligent, and aggregative DL objects that are capable of enforcing their own terms and conditions, negotiating access, and displaying their contents.

We refer to the above as the Smart Objects, Dumb Archive (SODA) model. Much of the traditional functionality associated with archives (terms and conditions, content display, etc.) has been "pushed down" into the objects, making the objects "smarter" and the archives "dumber". To demonstrate a SODA DL, we have a reference implementation, NCSTRL+, which implements each of the 3 strata listed above using the Dienst protocol and http services. The DLSs are provided by using the basic core of Dienst for searching, browsing and similar services. The archive functionality was originally implemented using a modified version of Dienst, but we are now transitioning to a simpler archive system. Our implementation of smart objects is "buckets", a special case of digital objects designed for DL use.

The observation that motivates the SODA model for DLs is that digital objects are more important than the archives that hold them. Many DL systems and protocols are reaching a level of complexity where DL interoperability and object mobility are hindered by the complexity of the archives that hold the objects. Our goal is to increase the responsibilities of objects, and decrease the responsibilities of archives. If digital objects themselves handle presentation,

terms and conditions and their own data management, it will be easier to achieve interoperability between heterogeneous DLs as well as increase object mobility and longevity. As a consequence, more DLSPs should be encouraged to build digital libraries for various user communities.

Note that digital libraries are used for user discovery of objects. Once the object has been found, the user interacts directly with the object itself. Archives exist primarily to assist DLs in locating objects -- they are generally not for direct user access. It is our belief that many digital libraries and the associated access protocols (e.g., Dienst [3], Repository Access Protocol (RAP) [10]) have become unnecessarily complex. We feel that the archived objects, not archives, should be responsible for the enforcement of terms and conditions, negotiation and presentation of content, etc. Although we expect some archive implementations to retain portions of the above functionality -- indeed, SOSA (Smart Objects, Smart Archives) may become the most desirable DL model -- we present a "dumb archive" to illustrate the full application of smart objects (buckets). When archives become "smart" again, it will be in other functionalities, not in duplication of bucket functionality. Using this terminology, Table 1 illustrates how the archive design space partions.

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

Table 1: Archive Design Space

Publishing in the SODA Model

Separating the functionality of the archive from that of the DLS 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. 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 (which is the set of the NASA buckets, the NASA archive(s), the NASA DLS, 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. 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 2 illustrates the SODA publishing model.

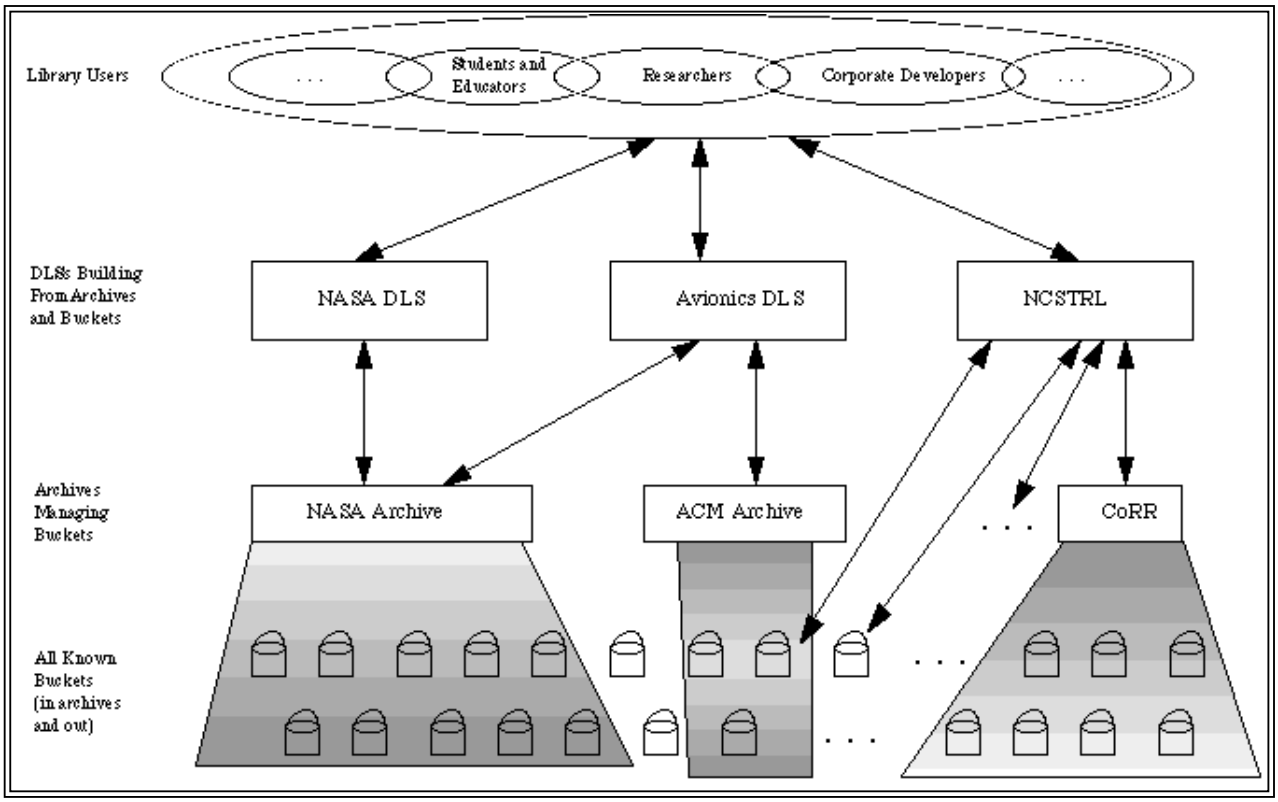

Figure 2: The SODA Publishing Model

The metadata implications are that buckets are the canonical source for their metadata. If a digital library service provider wishes to index a bucket, it simply asks the bucket for its metadata:

http://dlib.cs.odu.edu:8000/test-bucket3/?method=metadata

Or the DLS asks a proxy service that has already asked the bucket for its metadata. The DLS can then index the metadata according to its own indexing rules. If the DLS wishes to receive the metadata in a different format, it can ask for that format in the metadata method. We are implementing a metadata translation service (mdt) [16] that handles the dynamic conversion of bucket metadata. Buckets keep the translations of their metadata in a write-through cache. Buckets modify their own metadata files when objects are added to or deleted from them.

Bucket Services

We have been involved with a number of high traffic production NASA DLs since 1994, including the Langley Technical Report Server [11], the NASA Technical Report Server [15], and the NACA Report Server [14]. One of our early findings was that DL users wanted more than just the traditional literature; they also desired to have the associated software, datasets, images, video, and other supporting material related to a project or study [20]. We term this the *Pyramid of Scientific and Technical Information (STI)* (Figure 3). We argue that the traditional journal article, the common element in many DL projects, is often a mere abstract of a larger body of STI, most of which is informally archived or not archived at all [4].

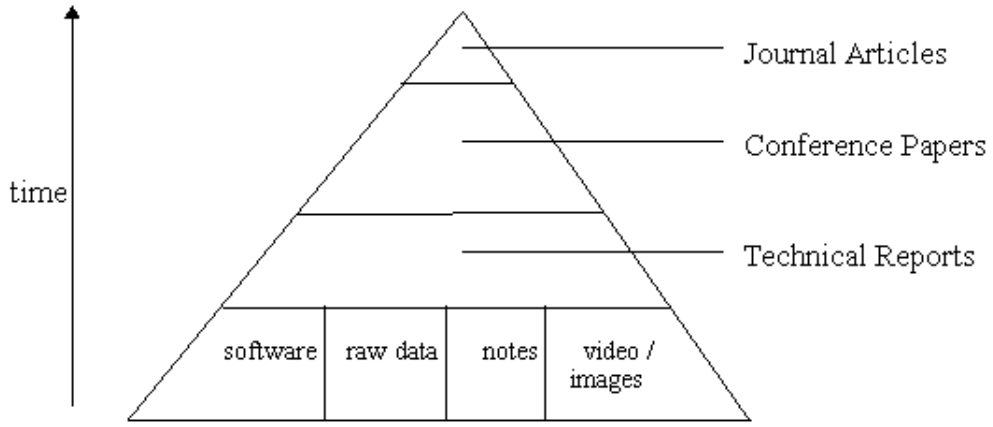

Figure 3: The Pyramid of STI

Another observation from NASA DL http log files is that 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.

Although we believe we have built attractive and useful interfaces for the NASA DLs, our main concern is that people have access to holdings and not that they use any given 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 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 DLs, but this leaves the researcher to re-integrate the information tuple by selecting pieces from multiple sources. 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. 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 [22] by providing a common mechanism to keep related STI products together. The current semantics of buckets include "elements", which are the unit of storage in buckets, and "packages", which are groups of elements. Figure 4 illustrates a typical bucket in a NASA DL application.

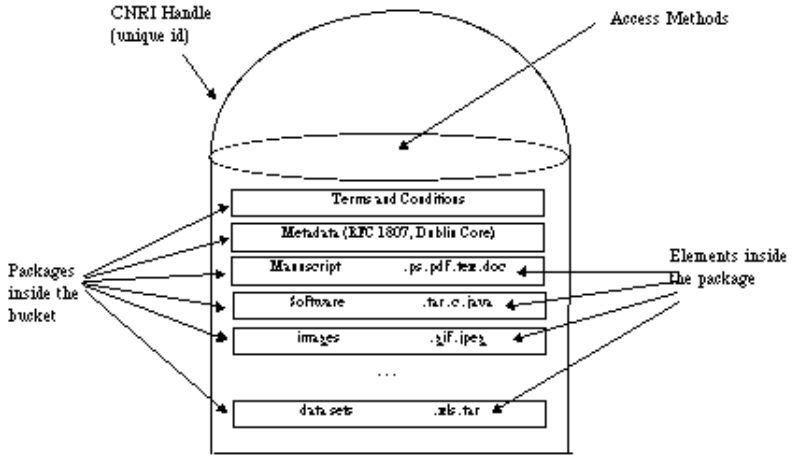

Figure 4: A Typical Bucket in a NASA DL

Our bucket prototypes are written in Perl 5, and make use of http as a transport protocol. Bucket metadata is stored in RFC-1807 format, and package and element information is stored in newly defined optional and repeatable fields. A bucket has all relevant files collected together using directories from file system semantics. Thus, an administrator can "cd" into the appropriate directory and access the contents. However, access for regular users occurs through the WWW. The bucket is accessible through a Common Gateway Interface (CGI) script that enforces terms and conditions, and negotiates presentation to the WWW client. The bucket presentation format is included in the bucket code, but we are currently planning to model presentation requirements using the Resource

Description Framework (RDF) [13] to provide a mechanism for providing dynamic presentation templates that can exploit known semantics during presentation. Table 2 provides a glimpse of bucket interaction. These methods are all invoked on a single test bucket filled with typical NASA data, but they could be invoked on any bucket. For example, all the methods in Table 2 could be invoked on:

http://www.cs.odu.edu/~dlibuser/nsf/nie/iri/

which is a bucket for the NSF project for the Interactive Remote Instruction (IRI) project. Buckets are further described in "Buckets: Aggregative, Intelligent Agents for Publishing" [17], including a discussion of their relation to Kahn-Wilensky Framework (KWF) Digital Objects (DOs) [5].

Method	Description
http://dlib.cs.odu.edu:8000/test-bucket3/?method=display or http://dlib.cs.odu.edu:8000/test-bucket3/	Displays the buck contents (default method)
http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_methods	Lists all the meth known by the buc
http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_principals	Lists defined principals (entries the password file) Access can be restricted to these principals.
http://dlib.cs.odu.edu:8000/test-bucket3/?method=metadata	Returns metadata the default format
http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_source or http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_source&target=display	Lists general sour code for the bucke (or for a particular method)
http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_logs	Lists the names of logs kept by the bucket
http://dlib.cs.odu.edu:8000/test-bucket3/?method=get_log&log=access.log	Displays the acces log for this bucket
http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_tc	Lists the terms an conditions for the bucket
http://dlib.cs.odu.edu:8000/test-bucket3/?method=display& pkg_name=appendix.pkg&element_name=NASA-95-tm4648-appendixA.html	Retrieves a restric element ("michael"/"micha to access)
http://dlib.cs.odu.edu:8000/test-bucket3/?method=display &redirect=http://babelfish.altavista.digital.com/cgi-bin/translate? urltext=http://dlib.cs.odu.edu:8000/test-bucket3/	An element as a reference to a 3rd party network serv (a translation serv in this example)

Table 2: Some Methods for Bucket Interaction

Archive Services

Our first implementation of a dumb archive was made by removing functionality from Dienst. 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 felt that tightly coupling the DL protocol with knowledge of individual file formats reduces the flexibility of the DL protocol.

We favor making Dienst less knowledgeable about dynamic topics such as file format, and making that 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 (Figure 5). In NCSTRL+, the final step of examining the directories structure is skipped, and the directory's index.cgi file is invoked. The default method for an index.cgi is generally the display method, so the user should notice little difference. However, at that point the bucket, not Dienst, determines what the user sees. This was our first dumb archive, a Dienst server that simply points to buckets rather than examining their contents.

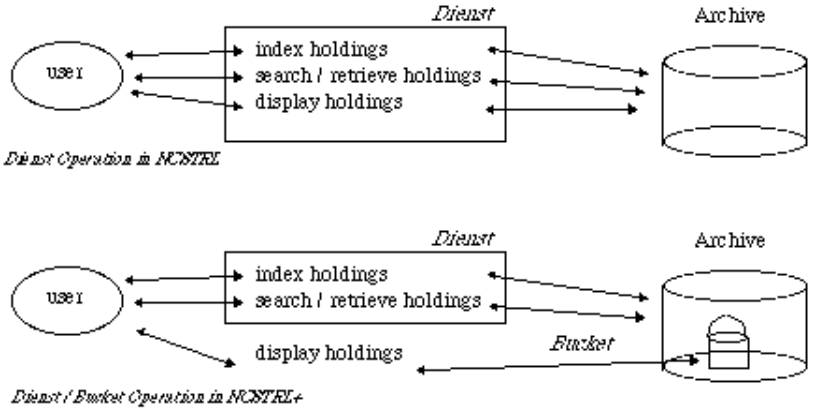

Figure 5: Shift of Responsibility in Dienst

We are currently working on an even simpler dumb archive, DA, which could operate with or without Dienst. This archive implements only a small number of functions (Table 3). DA is basically a set manager -- notice the DA has no search capabilities. The methods are currently being extended to support such things as arguments and conditional statements (i.e., "list all objects entered after December 12, 1995").

Method	Description
put	insert an item into the archive
delete	remove an item 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

Table 3: Dumb Archive Methods

To date, we have created 3 test archives: 1 for the NASA STI from LTRS, and 2 with sample NSF material. Table 4 illustrates the implemented DA methods for the LTRS archive as listed in machine readable format.

Method	URL
List All Buckets	http://www.cs.odu.edu/~dlibuser/ltrs/populate/testout/da.cgi/?method=list
List All Buckets Added By "mln"	http://www.cs.odu.edu/~dlibuser/ltrs/populate/testout/da.cgi/?method=list&originator
Information (metadata) about the archive	http://www.cs.odu.edu/~dlibuser/ltrs/populate/testout/da.cgi/?method=info

Table 4: DA Methods for the LTRS Archive

Digital Library Services

The NCSTRL+ project is based on the creation of buckets and the extension of the Dienst protocol. Dienst is a collection of DL "services" that receive messages encoded and transmitted via http. 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. We have defined clusters as a way of segregating search results by predefined metadata terms provided at the time of publishing. We have defined clusters for subject, archival type, terms and conditions, and language. Table 5 provides a list of searches of NCSTRL+, as well as the corresponding searches in LTRS. Although NCSTRL+ draws from the same corpus as LTRS, it is clear that NCSTRL+ is a richer, more expressive DLS.

Search	DL	URL
(default interface)	NCSTRL+	http://dlib.cs.odu.edu:8000/ncstrlplus.html
(default interface)	LTRS	http://techreports.larc.nasa.gov/ltrs/
computational fluid dynamics	NCSTRL+	http://dlib.cs.odu.edu:8000/Dienst/UI/2.0/Query/?author=&title=&abstract=computational+fluid+dynamics&authority=all&organization=ncstrlplus_subject=&ncstrlplus_archivaltype=&ncstrlplus_tc=&booleacluster=authority&sort=
computational fluid dynamics	LTRS	http://techreports.larc.nasa.gov/ltrs/ltrs.cgi?search_words=computational+AND+fluid+AND+dynamics

Table 5: NCSTRL+ Searches

With the NCSTRL+ searches, it is possible to re-cluster and re-sort the results pages. With LTRS, the search results once created are static. Experimentation with reclustering is the best way to observe its benefits.

Status and Future Work

Our future work follows two main thrusts: deployment of our current technology base into the NASA DLs, and the continued development of new technology. With respect to deployment of buckets and SODA, we are currently in the process of refining the aggregative operations of buckets, converting the entire LTRS corpus to buckets, and, most importantly, fine tuning the bucket toolset. Table 6 lists the current bucket tools (any username/password is sufficient). We are currently redesigning the interface for the Publishing Tool.

Tool	Description	Test URL
Publishing	creates and populates buckets	http://dlib.cs.odu.edu:8000/buckets/author.cgi
Management	controls the movement of buckets into archives	http://dlib.cs.odu.edu:8000/buckets/manage.cgi
Administration	used for update and other administration purposes	not suitable for demo at this time

Table 6: Bucket Tools

There are additional features in development for buckets as well. One is the inclusion of intelligence in the bucket so it can communicate with other buckets, people, and arbitrary network services. Elevating the status of buckets to computational and communicative entities creates a number of possible DL applications. The Bucket Matching System (BMS) will allow for buckets to search for possible "matches" and related works off-line. Buckets functionalities will include among others: format conversion of their contents, aging their contents, and updating their contents.

Another extension is the addition of customized viewers for buckets. Currently buckets will reveal their contents with the display method. However, we are working on buckets that maintain their metadata in extensible markup language (XML) and bucket-aware browsers can ask the bucket for their XML metadata, and then create a custom display (based on the user's preferences) from the bucket contents. The data remains inside the bucket, subject to the bucket's terms and conditions, but its structure is being viewed by an arbitrary and independent client. This work is currently being done within the context of bucket use in undergraduate education.

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 to dumb archives.

Bucket-like Projects

Buckets are most similar to the digital objects first described in the Kahn/Wilensky Framework [5], and its derivatives such as the Warwick Framework containers [9] and the more recent Flexible and Extensible Digital Object Repository Architecture (FEDORA) [1]. 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 relationship between buckets and KWF DOs is discussed further in [17]. FEDORA has not been completely implemented at this point, and it is unknown what repository or digital library protocol limitations will be present. Also, it is unknown if FEDORA plans to allow DOs to be intelligent agents.

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 [7], which are designed to allow for unlimited distribution of digital objects, but controlled access to their contents. Similarly, DigiBox 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" [21]. 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) [6] 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 [23]. 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 limit the usefulness of HDF as a generalized DL application. It is worth noting, however, that buckets of HDF files are entirely possible and appropriate.

Dumb Archive-like Projects

DA is interesting because of what it leaves out, not what it 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) [10] reveals many of 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 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 group have proposed a more recent service, the Collection Service [8]. 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 DA. When the Collection Service is available for testing, it is a good candidate to implement a SOSA model DL.

Conclusions

We did not set out to prove that interoperability is necessary nor that digital libraries are desirable, but rather we took these two statements as axiomatic. Neither did we set out to prove that our approach to interoperability is the right one. We did want to show that the approach is feasible, and we did want to have an implementation in order to begin experimentation with a large community and a large corpus. In this paper, we have described the approach and the

implementation. We believe we have made a case that pushing the intelligence and standards down to the object is the right model because that is where the ultimate information gain takes place for a user. The user's intention is to interact with the object, not with higher level library services -- these are just means to an end.

References

[1] R. Daniel & C. Lagoze, "Distributed Active Relationships in the Warwick Framework," Proceedings of the 2nd IEEE Metadata Conference, Silver Spring, MD, September 16-17, 1997.

[2] J. Davis & C. Lagoze, "The Networked Computer Science Technical Report Library," Cornell CS TR96-1595, July 1996.
http://cs-tr.cs.cornell.edu:80/Dienst/UI/1.0/Display/ncstrl.cornell/TR96-1595

[3] J. R. Davis, D. B. Krafft, & C. Lagoze, "Dienst: Building a Production Technical Report Server," Advances in Digital Libraries, Springer-Verlag, 1995, pp . 211-222.

[4] S. L. Esler & M. L. Nelson, "Evolution of Scientific and Technical Information Distribution," Journal of the American Society of Information Science, 49(1), 1998, pp. 82-91.
http://techreports.larc.nasa.gov/ltrs/PDF/1998/jp/NASA-98-jasis-sle.pdf

[5] R. Kahn & R. Wilensky, "A Framework for Distributed Digital Object Services," cnri.dlib/tn95-01, May, 1995.
http://www.cnri.reston.va.us/cstr/arch/k-w.html

[6] 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.

[7] 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

[8] C. Lagoze & D. Fielding, "Defining Collections in Distributed Digital Libraries." D-Lib Magazine, November 1998.
http://www.dlib.org/dlib/november98/lagoze/11lagoze.html

[9] 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/2.0/Describe/ncstrl.cornell/TR96-1593

[10] C. Lagoze, R. McGrath, E. Overly & N. Yeager, "A Design for Inter-Operable Secure Object Stores (ISOS)", Cornell CS TR95-1558, November 27, 1995.
http://cs-tr.cs.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell/TR95-1558

[11] Langley Technical Report Server
http://techreports.larc.nasa.gov/ltrs/

[12] 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

[13] E. Miller, "An Introduction to the Resource Description Framework," D-Lib Magazine, May

1998.
http://www.dlib.org/dlib/may98/miller/05miller.html

[14] NACA Technical Report Server
http://naca.larc.nasa.gov

[15] NASA Technical Report Server
http://techreports.larc.nasa.gov/cgi-bin/NTRS

[16] M. L. Nelson, K. Maly, D. R. Croom, Jr., & S. W. Robbins, "Metadata and Buckets in the Smart Object, Dumb Archive (SODA) Model," Proceedings of IEEE Meta-Data 99, Bethesda MD, April 6-7 1999.

[17] 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

[18] 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

[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] D. G. Roper, M. K. McCaskill, S. D. Holland, J. L. Walsh, M. L. Nelson, S. L. Adkins, M. Y. Ambur & M. Y. Ambur, "A Strategy for Electronic Dissemination of NASA Langley Technical Publications," NASA TM-109172, December 1994.
http://techreports.larc.nasa.gov/ltrs/PDF/tm109172.pdf

[21] 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.

[22] J. Sobieszczanski-Sobieski, "A Proposal: How to Improve NASA-Developed Computer Programs," NASA CP-10159, 1994, pp. 58-61.

[23] I. Stern, "Scientific Data Format Information FAQ," 1995.
http://www.cv.nrao.edu/fits/traffic/scidataformats/faq.html

Top | Contents
Search | Author Index | Title Index | Monthly Issues
Previous Story | Next Story
Home| E-mail the Editor

DOI: 10.1045/march99-maly