Metadata and Buckets in the Smart Object, Dumb Archive (SODA) Model

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

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

Delwin R. Croom, Jr.
NASA Langley Research Center, MS 158, Hampton, VA 23681
d.r.croom@larc.nasa.gov

Steven W. Robbins
Computer Sciences Corporation
NASA Langley Research Center, MS 157D, Hampton, VA 23681
s.w.robbins@larc.nasa.gov

ABSTRACT

We present the Smart Object, Dumb Archive (SODA) model for digital libraries (DLs), and discuss the role of metadata in SODA. The premise of the SODA model is to "push down" many of the functionalities generally associated with archives into the data objects themselves. Thus the data objects become "smarter", and the archives "dumber". In the SODA model, archives become primarily set managers, and the objects themselves negotiate and handle presentation, enforce terms and conditions, and perform data content management. Buckets are our implementation of smart objects, and da is our reference implementation for dumb archives. We also present our approach to metadata translation for buckets.

1.0 INTRODUCTION

The observation that motivated the Smart Object, Dumb Archive (SODA) model for digital libraries (DLs) is that data 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 data 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.

Since there is little consensus about terminology in the DL community, it is necessary to define the terms as used in this paper:

- *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), or more sophisticated objects such as buckets.

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 not generally for direct user access.

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 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 that it is necessary to evolve the notion of the object and 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. This will allow for objects to be represented in many DLs, and will also make the objects more resilient to DL system and protocol upgrades. If we break the monolithic structure of DLS-Archive-Object and define their interaction though simple interfaces, then all three are free to evolve independently. For example, special functionality data objects can be developed without impacting search engines, and archival software functionality can be extended without converting the data objects.

One of the results of moving functionality into the data objects is that the data objects become the canonical source for metadata about the objects. In our experience with designing, building and maintaining DLs [7], one significant problem is keeping metadata and data synchronized, especially preventing "drift" over time. Additionally, the linkage between metadata and data is often one-way: given the metadata, you can generally locate the data, but given the data, it is often difficult to locate or derive the corresponding metadata. Establishing a two-way mapping between metadata and data greatly facilitates DL interoperability, easily providing metadata to other DLs upon request.

NASA Langley Research Center and Old Dominion University are participating in a joint research program developing digital library (DL) technologies with particular emphasis on NASA’s Scientific and Technical Information (STI) Program [9]. In particular, the NCSTRL+ testbed is based on the Dienst DL protocol [1] and implements two new DL technologies: clusters and buckets. Clusters allow for the arbitrary partitioning of collections. We have defined five clusters for use in NCSTRL+:

1. Subject Category (computer science, physics, etc.)
2. Archival Type (pre-print, technical report, software, etc.)
3. Terms and Conditions (free, membership required, etc.)
4. Source Language (English, German, etc.)
5. Publishing Institution (included by default in Dienst)

Clusters are implemented as newly defined fields in the metadata. Unlike Title, Authors, and other

fields, clusters are only searchable through a controlled vocabulary presented by the user interface. Clusters and the NCSTRL+ project itself are more fully discussed in [11]. Buckets are the "smart objects" that we have developed for SODA.

2.0 BUCKETS

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 (images, video, 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 [13] by providing a common mechanism to keep all 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 1 illustrates a typical bucket in a NASA DL application.

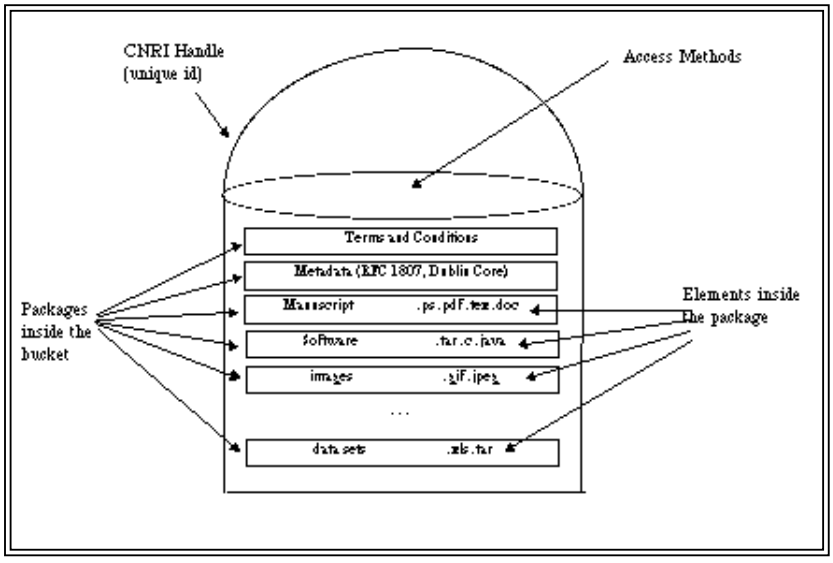

Figure 1: An Example Bucket

Our bucket prototypes are written in Perl 5, and make use of the fact that Dienst uses hypertext transfer protocol (HTTP) as a transport protocol. Like Dienst, bucket metadata is stored in RFC-1807 format [5], 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. Thus a Dienst 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 currently included in the bucket code, but we are currently planning to model presentation requirements using the Resource Description Framework (RDF) [8] 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.

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. 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. Table 1 provides a glimpse of bucket interaction. Buckets are further described in [12], including a discussion of their relation to Kahn-Wilensky Digital Objects [2].

Method	Description
http://dlib.cs.odu.edu:8000/test-bucket3/?method=display or http://dlib.cs.odu.edu:8000/test-bucket3/	Displays the bucket contents (default method)
http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_methods	Lists all the methods known by the bucket
http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_principals	List defined principals (entries in the password file). Access can be restricted to these principals.
http://dlib.cs.odu.edu:8000/test-bucket3/?method=metadata	Returns metadata in 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 source code for the bucket (or for a particular method)
http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_logs	Lists the names of all logs kept by the bucket
http://dlib.cs.odu.edu:8000/test-bucket3/?method=get_log&log=access.log	Display the access log for this bucket
http://dlib.cs.odu.edu:8000/test-bucket3/?method=list_tc	Lists the terms and conditions for the bucket

Table 1: Some Methods for Bucket Interaction

3.0 DUMB ARCHIVES

The SODA model for digital libraries is based on the belief that many digital libraries and the associated access protocols (e.g., Dienst, Repository Access Protocol (RAP) [3]) 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. 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. However, we present a "dumb archive" to illustrate the full application of smart objects (buckets). If archives become "smart" again, it will be in other functionalities, not in duplication of bucket functionality. We are working on an archive that implements only the following operations:

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	redirect to a bucket’s URL or URN

Table 2: Dumb Archive Methods

These methods represent the thrust of the dumb archive. 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").

The terminology we employ is that buckets are data objects. Archives are sets of data objects. Digital libraries are collections of services that users interact with to perform common tasks such as searching, browsing, submitting, etc. on data objects. DLs can build their collections by polling different archives, or the data objects themselves. Buckets, archives, and DLs are all loosely coupled. Buckets can exist outside of archives and DLs. Archives can feed multiple DLs. Multiple DLs can poll a single archive to build their collections.

The DA is basically 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 of 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 possibly novel and unique methods for searching browsing. Another possibility might be 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 this publishing model.

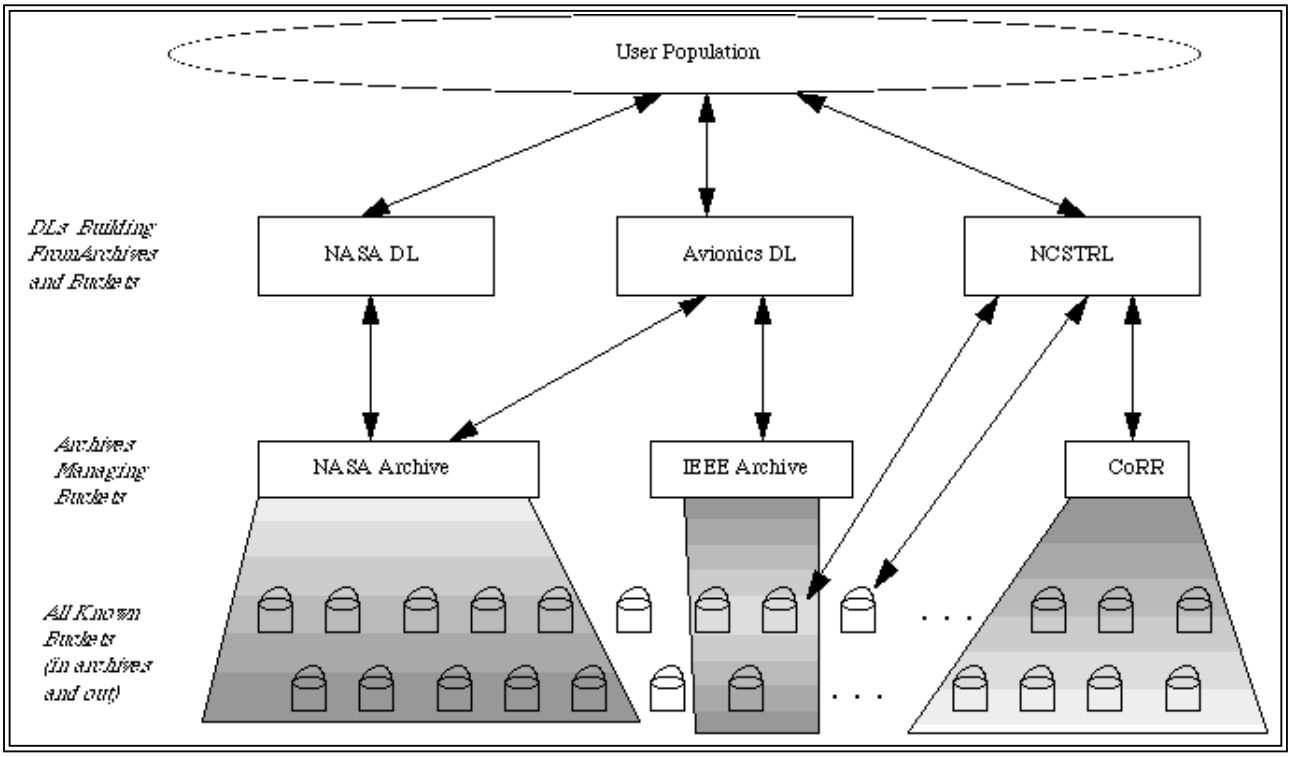

Figure 2: The SODA Publishing Model

The metadata implications are that buckets are the canonical source for their metadata. If a digital library 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 DL asks a proxy service that has already asked the bucket for its metadata. The DL can then index the metadata according to its own indexing rules. If the DL 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) 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.

4.0 METADATA TRANSLATION

Buckets currently store their metadata in RFC-1807 format. However, it is desirable for buckets to convert their metadata and be able to respond to metadata requests in different formats. We have implemented a metadata translation tool (mdt) which can be used as a Unix command line utility, or as a standalone server. The usage for mdt is:

```
mdt [-options] [<input-filename>] [<output-filename>]
```

Where:

```
options:
  -verbose: show some information about the translation
  -dublincore: output in Dublin Core metadata format
```

```
   -refer: output in refer metadata format
   -rfc1807: output in RFC-1807 metadata format
   -bibtex: output in Bibtex metadata format
   -server: operate in server mode
```

If <output-filename> is omitted, output is written to STDOUT; if <input-filename> is also omitted, input is read from STDIN and output is written to STDOUT.

The command line operation is useful for batch style bucket administration, but the server method is intended for buckets (or other applications) to use. Buckets can utilize mtd to store different versions of their metadata (either pre-converted, or using a write-through cache method). The following server transcripts illustrates the use of mdt in server mode while converting a refer [6] file to an RFC-1807 format:

```
lily> telnet dlib.cs.odu.edu 45811
Trying 128.82.4.76...
Connected to egbert.cs.odu.edu.
Escape character is '^]'.

OK-Metadata Translator v1.0
HELP

OK-Valid commands are:
  HELP   - Print (this) list of valid commands
  IN     - Begin sending metadata to translate; end with
           "." on line by itself
  OUT <format>
         - Output translated metadata in the specified <format>
  STATUS - Print current metadata and format
  QUIT   - End translation session

OK-Metadata Translator v1.0
IN

OK-Begin sending metadata; end with "." on line by itself
%T Accelerating Ground-Test Cycle Time:  The Six-Minute Model Change and Other V
%A Jerome T. Kegelman
%B 36th AIAA Aerospace Sciences Meeting and Exhibit
%C Reno, Nevada
%I NASA Langley Research Center, Hampton, VA  23681-0001
%D January 12-15, 1998
%R AIAA 98-0142
%K Langley Research Center; Cycle time management; Cycle time reduction; Test cy
%O (2MB)
%U-application/postscript ftp://techreports.larc.nasa.gov/pub/techreports/larc/1
%U-application/pdf http://techreports.larc.nasa.gov/ltrs/PDF/1998/aiaa/NASA-aiaa
%X The advantage of managing organizations to minimize product
development cycle time has been well established.  This paper provides
an overview of the wind tunnel testing cycle time reduction activities
at Langley Research Center (LaRC)...
.

OK-Received metadata; format = refer
STATUS

OK-Current metadata format is refer. Current metadata follows:
%T Accelerating Ground-Test Cycle Time:  The Six-Minute Model Change and Other V
%A Jerome T. Kegelman
%B 36th AIAA Aerospace Sciences Meeting and Exhibit
```

```
%C Reno, Nevada
%I NASA Langley Research Center, Hampton, VA  23681-0001
%D January 12-15, 1998
%R AIAA 98-0142
%K Langley Research Center; Cycle time management; Cycle time reduction; Test cy
%O (2MB)
%U-application/postscript ftp://techreports.larc.nasa.gov/pub/techreports/larc/1
%U-application/pdf http://techreports.larc.nasa.gov/ltrs/PDF/1998/aiaa/NASA-aiaa
%X The advantage of managing organizations to minimize product
development cycle time has been well established.  This paper provides
an overview of the wind tunnel testing cycle time reduction activities
at Langley Research Center (LaRC)...

OK-Metadata Translator v1.0
OUT
ERR-Unrecognized format. Recognized formats are refer, RFC-1807, Bibtex, and Dub
OK-Metadata Translator v1.0
OUT RFC-1807

OK-Translated metadata follows; format = RFC-1807:
BIB-VERSION:: X-NCSTRL+1.0
ID:: AIAA 98-0142
OTHER_ACCESS:: ftp://techreports.larc.nasa.gov/pub/techreports/larc/1998/aiaa/NA
OTHER_ACCESS:: http://techreports.larc.nasa.gov/ltrs/PDF/1998/aiaa/NASA-aiaa-98-
TITLE:: Accelerating Ground-Test Cycle Time: The Six-Minute Model Change and Oth
AUTHOR:: Jerome T. Kegelman
ABSTRACT:: The advantage of managing organizations to minimize
product development cycle time has been well established. This paper
provides an overview of the wind tunnel testing cycle time reduction
activities at Langley Research Center (LaRC)...
KEYWORD:: Langley Research Center
KEYWORD:: Cycle time management
KEYWORD:: Cycle time reduction
KEYWORD:: Test cycle time reduction
KEYWORD:: Productivity
KEYWORD:: Modern design of experiment
KEYWORD:: Telemetry
KEYWORD:: Quick disconnect
KEYWORD:: Cryogenic
ORGANIZATION:: NASA Langley Research Center, Hampton, VA 23681-0001
DATE:: January 1998
END:: AIAA 98-0142
.

OK-Metadata Translator v1.0
QUIT

OK-Bye
Connection closed by foreign host.
```

5.0 FUTURE WORK

We are in the process of converting some of NASA's digital libraries to use buckets. After that, we plan to convert the DLs to the full SODA model. We expect to have the Langley Technical Report Server (LTRS) [4] converted in the first part of 1999, with the NASA Technical Report Server (NTRS) [10] to follow. At this point, the NCSTRL+ project is converting the over 1600 items in the Langley Technical Report Server to buckets. The experience of mining the metadata of the LTRS library to create buckets is described in [7]. The collection of buckets created in that conversion process was 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 to enforce 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.

We are also in the process of extending buckets beyond aggregation, by also making them intelligent agents. We will implement a communications infrastructure that will allow buckets to communicate with other buckets as well as arbitrary network resources. The Bucket Matching System (BMS) will be a modular system that will allow for a variety of tasks, such as finding "matching" buckets, performing metadata scrubbing, posting content or internal updates for buckets, etc. We anticipate mdt functionality being moved into the BMS in the future, as this will be a common bucket communication requirement.

5.0 CONCLUSION

We feel that the SODA model will have significant implications in the digital library community. Shifting the responsibility of terms and conditions, presentation, and metadata maintenance to the archived object itself will allow for a greater heterogeneity of archived objects. Objects requiring special presentation, terms and conditions, or other considerations can now easily coexist with less specialized archived objects. Decoupling data objects, archives, and DL services such as searching and browsing should make it easier to expand and enrich these services independently in the future.

SODA promotes data objects as being more important than the data object’s archive. The objects are now the canonical source for its metadata as well as its data contents. We feel this emphasis and shift of functionality will enable additional progress to be made in digital libraries.

6.0 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/2.0/Describe/ncstrl.cornell%2fTR96-1595"

[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] C. Lagoze, R. McGrath, E. Overly and 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

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

[5] R. Lasher & D. Cohen, "A Format for Bibliographic Records," Internet RFC-1807, June 1995.

http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1807.txt

[6] M. E. Lesk, "Some Applications of Inverted Indexes on the UNIX System," Bell Laboratories Computing Science Technical Report # 69, 1978.

[7] K. Maly, S. N. T. Shen, M. Zubair & M. L. Nelson, "Generalizing an Existing Digital Library," Old Dominion University Computer Science Technical Report TR-99-01, February 1999.

[8] E. Miller, "An Introduction to the Resource Description Framework," D-Lib Magazine, May 1998.
http://www.dlib.org/dlib/may98/miller/05miller.html

[9] NASA Scientific and Technical Information Program
http://stipo.larc.nasa.gov/

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

[11] 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

[12] M. L. Nelson, K. Maly, S. N. T. Shen, & M. Zubair, "Buckets: Aggregative, Intelligent Agents for Publishing," NASA TM-1998-208419, May 1998.
http://techreports.larc.nasa.gov/ltrs/PDF/1998/tm/NASA-98-tm208419.pdf

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