Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries
(IEEE ADL '98), Santa Barbara, CA, April 22-24, 1998, pp. 128-136.

NCSTRL+: Adding Multi-Discipline and Multi-Genre Support to the Dienst Protocol Using Clusters and Buckets

Michael L. Nelson
NASA Langley Research Center, MS 158, Hampton, VA, 23681-0001
m.l.nelson@larc.nasa.gov

Kurt Maly, Stewart N. T. Shen, Mohammad Zubair
Old Dominion University Computer Science Department, Norfolk VA, 23529
{maly, shen, zubair}@cs.odu.edu

Abstract

We describe NCSTRL+, a unified, canonical digital library for scientific and technical information (STI). NCSTRL+ is based on the Networked Computer Science Technical Report Library (NCSTRL), a World Wide Web (WWW) accessible digital library (DL) that provides access to over 100 university departments and laboratories. NCSTRL+ implements two new technologies: cluster functionality and publishing "buckets". We have extended Dienst, the protocol underlying NCSTRL, to provide the ability to "cluster" independent collections into a logically centralized digital library based upon subject category classification, type of organization, and genres of material. The bucket construct provides a mechanism for publishing and managing logically linked entities with multiple data forms as a single object. The NCSTRL+ prototype DL contains the holdings of NCSTRL and the NASA Technical Report Server (NTRS). The prototype demonstrates the feasibility of publishing into a multi-cluster DL, searching across clusters, and storing and presenting buckets of information.

1. Introduction

Digital libraries (DLs) are an important research topic in many scientific communities and have already become an integral part of the research process. However, access to these DLs is not as easy as users would like. Digital library projects are partitioned by both the discipline they serve (computer science, aeronautics, physics, etc.) and by the format of their holdings (technical reports, video, software, etc.). A recent survey found over 10 existing or recent different World Wide Web (WWW) oriented digital library projects spanning over a range of different disciplines [6]. In short, each community is hand crafting their own digital library infrastructure.

The convergence of several technologies and importance of cross-disciplinary research is necessitating a need for multi-disciplinary DLs. There are two significant problems with current DLs. First, interdisciplinary research is difficult because the collective knowledge of each discipline is stored in incompatible DLs that are known only to the specialists in the subject. The second significant problem is that although scientific and technical information (STI) consists of manuscripts, software, datasets, etc., the manuscript receives the majority of attention, and the other components are often discarded [21]. Although non-manuscript digital libraries such as the software archive *Netlib* [4] have been in use for some time, they still place the burden of STI reintegration on the customer. A NASA study found that customers desire to have the entire set of manuscripts, software, data, etc. available in one place [20]. With the increasing availability of all-digital storage and transmission, the re-integration of the STI output back to its original state is possible.

Old Dominion University and NASA Langley Research Center are developing NCSTRL+ to address the multi-discipline and multi-genre problems. NCSTRL+ is based on the Networked Computer Science Technical Report Library (NCSTRL) [5], which is a highly successful digital library offering access to over 100 university departments and laboratories since 1994, and is implemented using the Dienst protocol [11]. During the development stage, NCSTRL+ includes selected holdings from the

NASA Technical Report Server (NTRS) [16] and NCSTRL, providing *clusters* of collections along the dimension of disciplines such as aeronautics, space science, mathematics, computer science, and physics, as well as clusters along the dimension of publishing organization and genre, such as project reports, journal articles, theses, etc. NCSTRL+ holdings will be published in *buckets* [18], an object-oriented construct for creating and managing collections of logically related information units as a single object. A bucket can contain both different data syntax (PostScript, PDF, Word, etc.) and different data semantics (manuscripts, data files, images, software, etc.)

2. Background

The core technology for NCSTRL+ has existed for several years. In 1992, the ARPA-funded CS-TR project [8] began as did the Langley Technical Report Server (LTRS) [17]. In 1993, the Wide Area Technical Report Server (WATERS) [14] shared a code base with LTRS. In 1994, LTRS launched the NTRS [16], and the CS-TR and WATERS projects formed the basis for the current NCSTRL. In 1997, NTRS and NCSTRL formed the basis for NCSTRL+.

We chose to implement NCSTRL+ using Dienst instead of other digital library protocols such as TRSkit [19] because of Dienst's success in several years of production in NCSTRL. Dienst appears to be the most scalable, flexible, and extensible of digital library systems we surveyed [6]. Dienst also serves as the basis for other focused digital library projects, including: the Electronic Thesis and Dissertation Project [7], the University of Virginia undergraduate engineering thesis project [22] and the ACM SIGIR conference proceedings project (which requires ACM authentication) [1].

Our buckets are similar in concept to the "digital objects" first proposed in [10]. It is important to note that many services have had "proto-buckets" in operation for some time. However, they provide only different formats of a single manuscript, or may support the concept of separate pages within a manuscript. They do not support an interface to a collection of related objects such as the manuscript, software, datasets, etc. We chose the term "buckets" because related terms such as "objects", "packages" and "containers" are greatly overloaded in the computer science realm and because "buckets" provide a clear visual metaphor for the concept when speaking with non-computer scientists.

There are other projects that are modifying Dienst to support functionality similar to clustering. ArquiTec [3] is a project of the Portuguese National Library. Among some additional features, ArquiTec adds the distinction of "official", "formal" and "informal" documents to Dienst which is somewhat similar to our definition of clusters. Also, the MeDoc project [2] has built in support in Dienst for hierarchical partitioning of authoring institutions and partitioning along the language of the reports (e.g., English, German, etc.). Both the ArquiTec and MeDoc projects add cluster-like functionalities, but neither are generalized for the purposes of subject, genre and organization as we have defined them.

3. Clusters of Dienst Servers

While Dienst is a successful production quality DL protocol, it has some inherent limitations that prevent additional features from being added. Among these is the inability to subdivide collections along anything other than institutional boundaries.

3.1 Overview of Dienst

While Dienst is discipline independent, it is currently discipline monolithic. It makes no provision for knowledge of multiple subjects within its system. While it is possible to set up a collection of Dienst servers independent of NCSTRL, there is no provision for linking such collections of servers into a higher level meta-library. Dienst consists of 5 components: 1) Repository Service; 2) Index Service; 3) Meta-Service; 4) User Interface Service; and 5) Library Management Service. Each of the services has a list of valid "verbs" that the service understands, and some of the verbs can take arguments. Dienst uses the hypertext transfer protocol (HTTP) as a transport protocol. The standard format is:

http://host:port/Dienst/Service/Version/Verb/Arguments

An example of a valid Dienst request is:

http://repository.larc.nasa.gov:8080/Dienst/UI/2.0/Search

This contacts the Meta-Server service at repository.larc.nasa.gov and requests a list of publishing authorities that this machine contains. Dienst names objects in collections using the CNRI Handle system [9]. We are using the experimental and unregistered handles of "ncstrlplus.larc" and "ncstrplus.odu.cs". Meta-data for objects is stored in the RFC-1807 format [13].

The basic architecture of NCSTRL has a single entry point ("home page") for user access. Each publishing authority (in practice, an authority generally corresponds to a university department or laboratory) runs its own copy of the Dienst software. The home page gathers the queries and dispenses the queries in parallel to each server, gathers the results, and displays the correlated results to the user. To assist with performance and reliability, Dienst now employs a Regional Meta-Server (RMS) to partition all NCSTRL participants into geographic regions. The various RMSs share their data with the Master Meta Server (MSS) at Cornell (the home of Dienst and NCSTRL). A Merged Index Server (MIS) provides a single index of all the metadata outside a region. A search query is sent to all standard sites within a region, and to the region's MIS for metadata outside the region.

3.2 Modifications to Dienst

Clusters are a way of aggregating logically grouped sub-collections in a DL along some criteria. NCSTRL+ provides 3 clusters: organization, data genre, and subject category. *Genre* is a term provided by E. Fox in a private communication and refers to distinguishing between journal articles, technical reports, theses and dissertations, etc.

Dienst currently carries no concept of subject category in its protocol, despite having provisions for specifying keywords from the title, authors, and abstract. In fact, digital libraries using the Dienst protocol such as NCSTRL have the implicit assumption that all holdings are computer science related. We have modified some of the Dienst verbs to take 2 new fields to specify the clusters, ncstrlplus_sti_topics and ncstrlplus_search_genres. We use the already defined authority field for the organization cluster. These new fields were our initial attempt to demonstrate cluster functionality to the user. The term "clusters" for this purpose is due to C. Lagoze, who in a private communication proposed a new Dienst service (for a total of six Dienst services), a separate cluster service allowing the creation of clusters of Dienst servers along arbitrary criteria. If this new cluster service is added to a future version of Dienst, then our modifications will not be used in a production release of Dienst. Minimizing the number of source code modifications necessary was a high priority and we have been able to limit our changes to less than 10 of the nearly 150 files in the Dienst 4.1.8 distribution.

A separate service may not be necessary to provide a robust cluster functionality. An alternative to providing a new service is to provide cluster arguments to the appropriate message verbs in existing clusters. Table 1 lists the proposed verb modifications to provide clusters without a separate cluster service. However, the final decision on the direction of the production version of Dienst will reside with the NCSTRL Steering and Working Groups.

Table 1. Proposed Cluster Arguments to Verbs

Service	Message Verb	Argument	Argument Type
Index	List-Contents	cluster=	optional
Index	SearchBoolean	cluster=	optional
Meta	Publishers	cluster=	optional
Meta	Indices	cluster=	optional
Meta	Repositories	cluster=	optional
Meta	Lite	cluster=	optional
UI	Search	none	N/A
UI	QueryNF	cluster=	optional
UI	BrowseYears	cluster=	optional
UI	ListYears	cluster=	optional
UI	BrowseAuthors	cluster=	optional
UI	ListAuthors	cluster=	optional
LibMgt	ListClusters (proposed)	none	N/A
LibMgt	DescribeClusters (proposed)	none	N/A

For the NCSTRL+ prototype, we adopted the NASA STI subject categories. A full listing of the subject categories can be found in [15]. The NASA STI topics are attractive since they are familiar to the majority of our customer base, and they also provide over 100 subtopics while producing only a small number of high level topics (Table 2). The NASA STI topics have a decidedly aerospace slant, but they have a reasonable description of other disciplines, and appeared to be more general than similar listings from places such as the Defense Technical Information Center (DTIC). Most professional society's cataloging schemes are too focused on their specific discipline to provide the general framework for NCSTRL+.

Table 2. NASA STI Main Topics

Subject Category	Code
Aeronautics	01
Astronautics	12
Chemistry and Materials	23
Engineering	31
Geosciences	42
Life Sciences	51
Mathematical and Computer Sciences	59
Physics	70
Social Sciences	80
Space Sciences	88

NCSTRL+ reads its known subject and genre categories from preference files, so future augmentation or replacement of this list should not be difficult. The NASA STI topics are not meant to replace an institution's use of any subject specific categories, such as the ACM CR categories. Rather, NCSTRL+ will maintain a mapping of how various specialized classification schemes map into the larger NASA STI topics . The NASA STI topics for NCSTRL+ are implemented as a new optional and repeatable field in RFC-1807 format.

Table 3 shows the currently defined values for the genre cluster. These values are drawn from our own STI experience, unlike the NASA STI subject categories which are borrowed from an organization that is charged with maintaining such a list. Additional values for a usable set of genres across many disciplines may need to be created. In particular, the present genre "data collections" will likely be replaced by a set of more descriptive entries. Hierarchical entries, such as that present in the NASA STI subject categories, may be needed. Also, it may be desirable to have the ability for discipline-centric genre codes to co-exist with NCSTRL+ general genres, much like society-defined subject categories co-exist within the framework provided by the NASA STI subject categories. However, more experience is needed before implementation recommendations can be made.

Table 3. Defined Genres in NCSTRL+

Genre	Code
Courseware	1
Agency/Project Reports	2
Contractor Reports	3
Theses/Dissertations	4
Conference Papers	5
Journal Articles	6
Technical Reports	7
Books	8
Patents	9
Data Collections	10

4. 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 (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. A single bucket can have 0 or more *packages*. Packages can correspond to the semantics of the information (manuscript, software, etc.), or can be more abstract entities such as the metadata for the entire bucket, bucket terms and conditions, pointers to other buckets or packages, etc. A single package can have 1 or more *elements*, which are typically different file formats of the same information, such as the manuscript package having both PostScript and PDF elements. Packages and elements are illustrated in Figure 1.

4.1 Bucket Requirements

All buckets have unique ids (CNRI handles) associated with them. Buckets are of

arbitrary size, and there is no defined limit on the number of packages a bucket can contain, nor the number of elements a package can contain. Both packages and elements can be "pointers" to remote objects, including other buckets, bucket packages, or bucket elements.

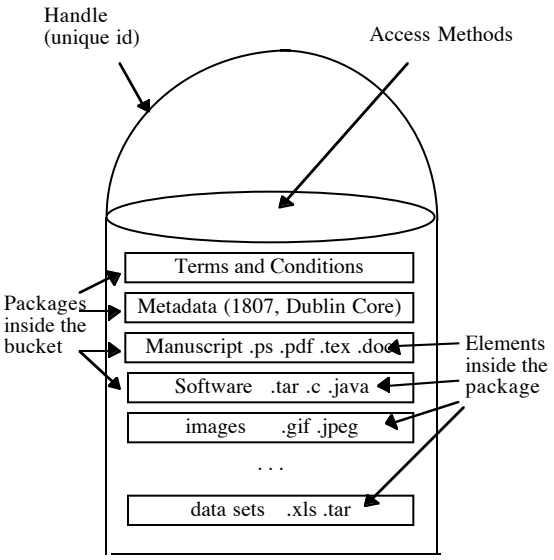

Figure 1: Bucket Architecture

Buckets are intended to be either standalone objects or to be placed in digital libraries. A standalone bucket can be accessible through normal WWW means without the aid of a repository. Buckets are intended to be useful even in repositories that are not knowledgeable about buckets in general, or possibly just not about the specific form of buckets. Buckets should not lose functionality when removed from their repository. The envisioned scenario is that NCSTRL+ will eventually have moderate numbers of (10s - 100s of thousands) of intelligent, custom buckets instead of large numbers (millions) of homogenous buckets. Traditionally, repositories contain all the intelligence and functionality for accessing its contents. Buckets provide the ability to shift some or all of the this intelligence to the archived object itself. This could be most useful when individual buckets require custom terms and conditions for access (security, payment, etc.).

Table 4 lists the required bucket methods; other methods can be custom defined. These are the only defined means of interaction with buckets. Note that Table 4 differs from protocols such as the Repository Access Protocol (RAP) [12] in that we have defined actions buckets perform on themselves, not actions a repository performs on buckets. Although the two are not mutually exclusive, the current plan is to not implement RAP for NCSTRL+.

Table 4. Required Bucket Methods

Method	Description
metadata	returns the bucket's metadata in its native form
display	default method; bucket "unveils" itself to requester
id	returns the bucket's unique identifier (handle)
tc	describes the nature of the bucket's terms and conditions
list_methods	list all methods known by a bucket
list_owners	list all principals that can modify the bucket
add_owner	add to the list of owners
delete_owner	delete from the list of owners
add_package	adds a package to an existing bucket
delete_package	deletes a package from an existing bucket
add_element	adds an element to an existing package
delete_element	deletes an element from an existing package
get_package	get all elements within a package
get_element	get a single element within a package
add_method	"teaches" a new method to an existing bucket
delete_method	removes a method from a bucket
copy_bucket	export a copy of a bucket, original remains
move_bucket	move the original bucket, no version remains

4.2 Bucket Tools

There are two main tools for bucket use. One is the *author tool* (Figure 2), which allows the author to construct a bucket with no programming knowledge. Here, the author specifies the metadata for the entire bucket, adds packages to bucket, adds elements to the packages, provides metadata for the packages, and selects applicable clusters. The author tool gathers the various packages into a single component and parses the packages based on rules defined at the author's site. Many of the options of the author tool will be set locally via the second bucket tool, the *management tool*. The management tool provides an interface to allow site managers to configure the default settings for all authors at that site. The management tool also provides an interface to query and update buckets at a given repository. Additional methods can be added to buckets residing in a repository by invoking the add_method on them and transmitting the new code. From this interface, the manager can halt the archive and perform operations on it, including updating or adding packages to individual buckets, updating or adding methods to

groups of buckets, and performing other archival management functions.

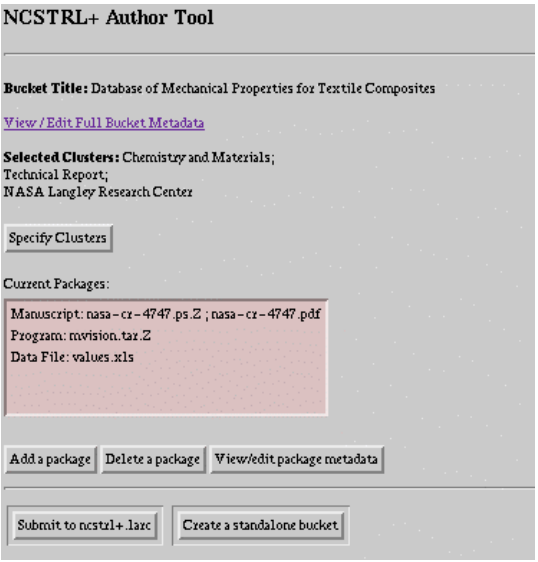

Figure 2: Author Tool

4.3 Bucket Implementation

In the previous section, we defined the requirements of bucket functionality independent of implementation. Our bucket prototypes are written in Perl 5, and make use of the fact that Dienst uses HTTP as a transport protocol. 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 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. To add a new file format, such as the increasingly popular PDF, Dienst configuration files have to be changed. If the protocol was resident only at one site, this would be acceptable. However, Dienst servers are running at nearly 100 sites – protocol additions require a coordinated logistical effort to synchronize versions and provide uniform capability.

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 bib meta data 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 Dienst is no longer determining what the user sees, the bucket is.

5. Using NCSTRL+

NCSTRL is successful in part because it is "easy" to use, both as a user searching the archive and as an author publishing into the archive. We must prevent the additional functionality of NCSTRL+ from significantly altering the usage profiles that NCSTRL users and contributors have come to expect.

5.1 Searching NCSTRL+

NCSTRL+ searching is similar to searching NCSTRL, with the addition of specifying desired clusters to search. Figure 3 shows the how the advanced fielded search form of NCSTRL+ is modified, allowing the selection of desired subject categories and data genres. A search results page includes the keyword and cluster hit results. The user will select the desired bucket from this page. At that point, the bucket will return the defined default initial interface of the bucket, which will be dependent on the bucket contents and the rules present. In practice, the bucket presentation will look largely similar to the choices available to current users of NCSTRL. This is especially true if the buckets in which they are interested only contain various manuscript formats. However, the real benefit is the richer presentation formats

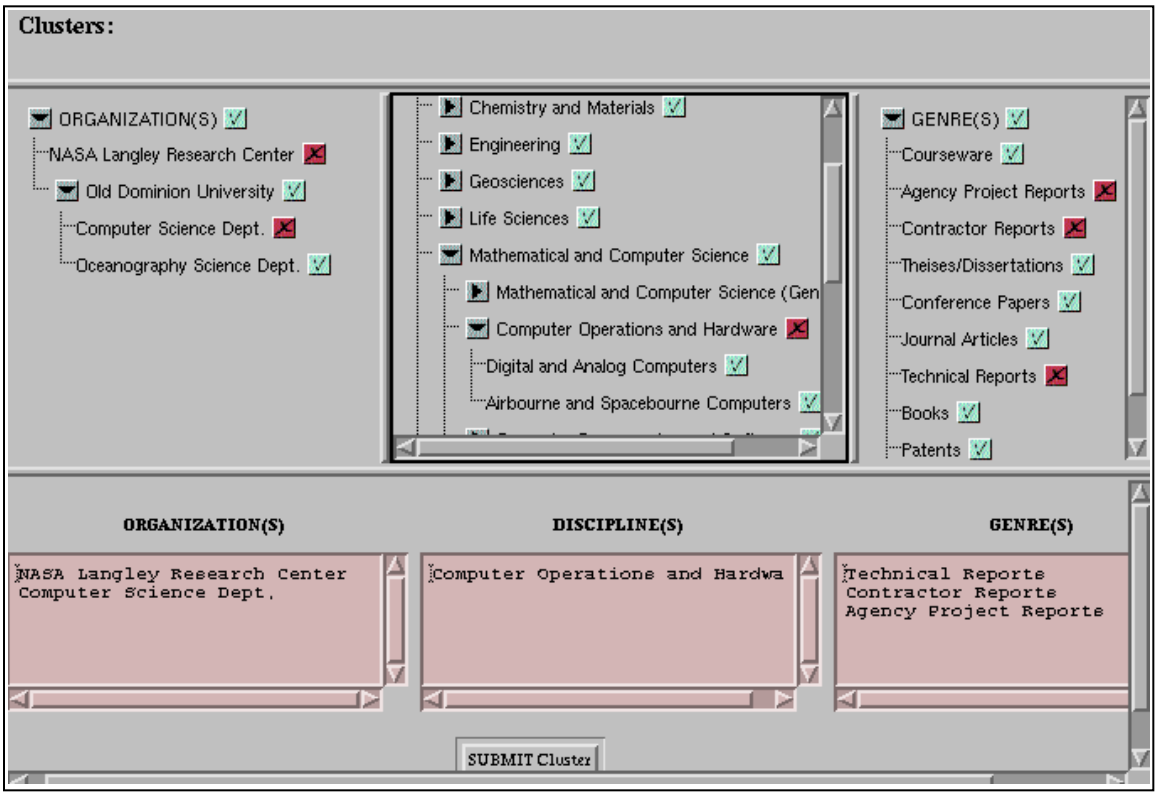

Figure 3: Cluster Specification in NCSTRL+

available if the bucket has non-manuscript packages. Our initial bucket interface is similar to NCSTRL, with the exception that the additional data semantics are presented (software, datasets, etc.).

5.2 Publishing into NCSTRL+

The goal of NCSTRL+ is to produce the least intrusive interface possible to the author. The authoring process for NCSTRL+ is to be as similar to authoring into NCSTRL as possible. Additions include the ability to add to a bucket multiple data semantics and formats through using multiple selection boxes to select local files. Publishing a manuscript in NCSTRL is equivalent to publishing a package in NCSTRL+, and publishing a bucket is the sum of publishing all of its packages. The author also has to choose the appropriate cluster to place the new bucket in. This step can be skipped if the site manager has defined a default, or if authors have saved a value already in their preferences.

6. Future Work

We are using the author tool to populate NCSTRL+ to gain insight on how to improve its operation. We are starting with buckets authored at Old Dominion University and NASA Langley Research Center and are choosing the initial entries to be "full" buckets, with special emphasis on buckets relating to NSF projects for ODU and for windtunnel and other experimental data for NASA. Until NCSTRL+ becomes a full production system, we are primarily seeking rich functionality buckets that contain diverse sets of packages.

It is also important to note that adding a subject category mechanism to NCSTRL+ provides the necessary groundwork for additional services for digital libraries using Dienst. These could include subject-based browsing of NCSTRL+ holdings, as well as selected dissemination of information (SDI). This would be most useful if users were offered a subscription option to receive digested updates (i.e., e-mail messages) of new additions to NCSTRL+ in specified subject areas. The initial defined subject categories for NCSTRL+ and cross-listing them with other subject-specific

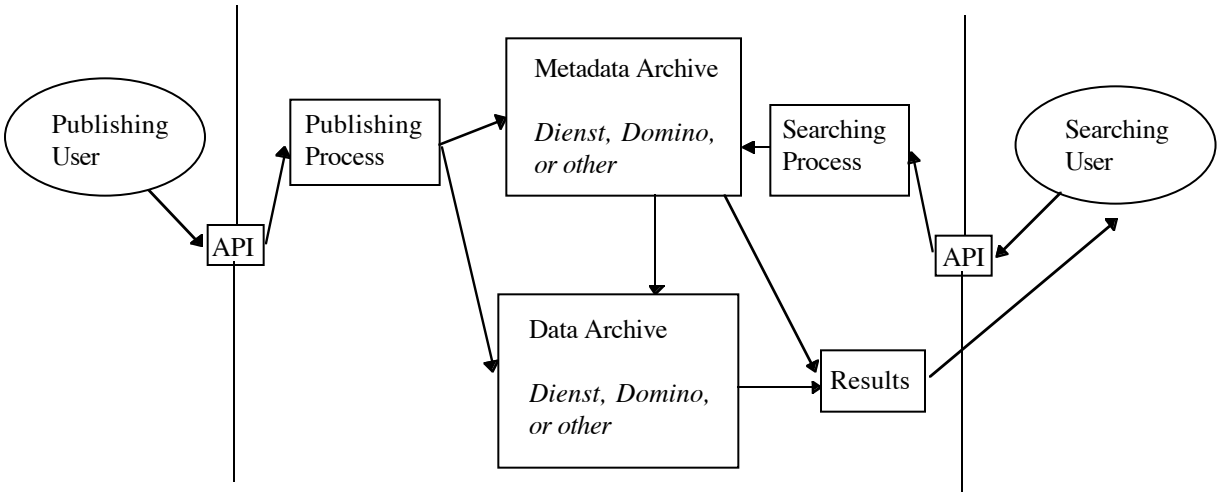

Figure 4. General NCSTRL+ Architecture Without Dienst Dependencies

categorization schemes is intended to provide a working framework for evaluating the prototype. As more experience in NCSTRL+'s use is gained, the fine tuning of the subject categories and appropriate cross listing becomes an area that would benefit from the attention of a professional cataloger.

We are also planning to implement buckets using Lotus Notes and Domino in addition to the current CGI and Perl implementation. The bucket API as defined in Table 5 will remain unchanged. In experimenting with Notes and Domino, we also plan to investigate implementing NCSTRL+ components without using Dienst. We plan to evolve NCSTRL+ to support a generalized publishing and searching model that can be implemented using Dienst or other DL protocols. Figure 4 illustrates the generalized DL architecture.

7. Conclusions

To meet the increased requirements for multi-disciplinary activities in educational and scientific communities, we have prototypes of NCSTRL+ and are in the process of full implementation. The most significant technology from this project is the concept of buckets as a construct to capture multiple data formats and genres in an intuitive manner. NCSTRL+ provides a platform for experimentation for testing user response to multi-discipline clusters and logical collections of STI. We are in the process of experimenting with users at NASA and Old Dominion University.

From the users' perspective, the publishing and searching interfaces are largely unchanged. However, it is unknown what impact the cluster and bucket modifications have on network load, search and retrieval times, the users' perceived quality of searching multiple clusters, etc. To determine these unknowns, NCSTRL+ will have to grow to a large enough size to be considered a useful production system. The authors seek other users and participants for NCSTRL+.

8. References

[1] ACM SIGIR On-Line Conference Proceedings, http://turing.acm.org:8071/

[2] S. Adler, U. Berger, A. Bruggemann-Klein, C. Haber, W. Lamersdorf, M. Munke, S. Rucker, & H. Spahn, "An Electronic Library for Grey Literature based on NCSTRL," 1997. http://medoc.informatik.unihamburg.de/Dienst/htdocs/dagstuhl/dagstuhl.ps

[3] J. L. Borbinha, J. Ferreira, J. Jorge, & J. Delgado, "A Digital Library for a Virtual Organization," *Proceedings of the 31st Hawaii International Conference on Systems Science (HICSS-31),* January 6-9, 1998.

[4] S. Browne, J. Dongarra, E. Grosse, S. Green, K. Moore, T. Rowan, & R. Wade, "Netlib Services and Resources," University of Tennesse Technical Report UT-CS-93-222, 1993.

[5] 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.

[6] S. L. Esler & M. L. Nelson, “The Evolution of Scientific and Technical Information Distribution,” *Journal of the American Society of Information Science,* 49(1), 1998, pp. 82-91.

[7] E. Fox, J. Eaton, G. McMillan, N. Kipp, L. Weiss, E. Arce, & S. Guyer. “National Digital Library of Theses and Dissertations: A Scalable and Sustainable Approach to Unlock University Resources,” *D-Lib Magazine, The Magazine of Digital Library Research*, Sep. 1996. http://www.dlib.org/dlib/september96/theses/09fox.html

[8] R. Kahn, "An Introduction to the CS-TR Project,” December 1995. http://www.cnri.reston.va.us/home/describe.html

[9] R. Kahn, “The Handle System Version 3.0: An Overview.” http://www.handle.net/docs/overview.html

[10] 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/kw.html

[11] C. Lagoze, E. Shaw, J. R. Davis, & D. B. Krafft, “Dienst: Implementation Reference Manual,” Cornell Computer Science Technical Report TR95-1514, 1995.

[12] C. Lagoze & D. Ely, “Implementation Issues in an Open Architectural Framework for Digital Object Services,” Cornell University Computer Science Technical Report, TR95-1540, June, 1995.

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

[14] K. Maly, J. French, A. Selman, & E. Fox, “Wide Area Technical Report Service,” *Proceedings of the Second International World Wide Web Conference*, Chicago, IL, October 21-23, 1994, pp. 523-533.

[15] NASA Scientic and Technical Information Program, “NASA STI Topics.” ftp://ftp.sti.nasa.gov/pub/scan/SCAN-TOPICS

[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] M. L. Nelson, K. Maly & S. N. T. Shen, “Buckets, Clusters and Dienst,” Old Dominion University Computer Science Technical Report 97-30, July 1997. (Also available as NASA TM-11287)

[19] M. L. Nelson & S. L. Esler, “TRSkit: A Simple Digital Library Toolkit,” *Journal of Internet Cataloging*, 1(2), pp. 41-55.

[20] D. G. Roper, M. K. McCaskill, S. D. Holland, J. L. Walsh, M. L. Nelson, S. L. Adkins, M. Y. Ambur & B. A. Campbell, “A Strategy for Electronic Dissemination of NASA Langley Technical Publications,” NASA TM-109172, December 1994.

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

[22] UVa SEAS Electronic Undergraduate Thesis Pilot, http://univac.cs.virginia.edu:3066/SEAS_ETD.html