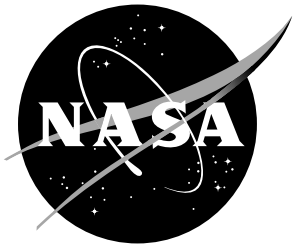
NASA Technical Memorandum 112876



Building Multi-Discipline, Multi-Format Digital Libraries Using Clusters and Buckets

Michael L. Nelson
Langley Research Center, Hampton, Virginia

August 1997

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23681-0001

BUILDING MULTI-DISCIPLINE, MULTI-FORMAT DIGITAL LIBRARIES USING CLUSTERS AND BUCKETS

by

Michael L. Nelson
B.S. May 1991, Virginia Polytechnic Institute and State University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

MASTER OF SCIENCE

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY

August 1997

ABSTRACT

BUILDING MULTI-DISCIPLINE, MULTI-FORMAT DIGITAL LIBRARIES
USING CLUSTERS AND BUCKETS

Michael L. Nelson
Old Dominion University, 1997
Co-Chairs of Advisory Committee: Dr. Kurt Maly
Dr. Stewart N. T. Shen

Our objective was to study the feasibility of extending the Dienst protocol to enable a multi-discipline, multi-format digital library. We implemented two new technologies: *cluster* functionality and publishing *buckets*. We have designed a possible implementation of clusters and buckets, and have prototyped some aspects of the resultant digital library.

Currently, digital libraries are segregated by the disciplines they serve (computer science, aeronautics, etc.), and by the format of their holdings (reports, software, datasets, etc.). NCSTRL+ is a multi-discipline, multi-format digital library (DL) prototype created to explore the feasibility of the design and implementation issues involved with created a unified, canonical scientific and technical information (STI) DL. NCSTRL+ is based on the Networked Computer Science Technical Report Library (NCSTRL), a World Wide Web (WWW) accessible DL that provides access to over 80 university departments and laboratories.

We have extended the Dienst protocol (version 4.1.8), the protocol underlying NCSTRL, to provide the ability to cluster independent collections into a logically

centralized DL based upon subject category classification, type of organization, and genre of material. The concept of buckets provides a mechanism for publishing and managing logically linked entities with multiple data formats. NCSTRL+ contains some of th 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. The clusters provide multi-discipline access, and the buckets allow for the storage and retrieval of new information formats, increasing the availability of all forms of STI. We found that the overhead for these additional capabilities is minimal to both the author and the user when compared to the equivalent process within NCSTRL. We cover the current modifications to Dienst, discuss the proposed modifications to Dienst that we are coordinating with the NCSTRL Working Group, discuss the proposed tools to facilitate the construction and use of buckets, and discuss additional capabilities for which clusters and buckets provide the foundation.

ACKNOWLEDGMENTS

Professors Kurt Maly and Stewart N. T. Shen provided the direct advisement for this research.

NASA Langley Research Center has provided me with the opportunity and resources to perform digital library research and development. In particular, I would like to thank Gretchen Gottlich, David Bianco, Sandra Esler and Ming-Hokng Maa for their support in earlier projects, and recently Del Croom for his support in developing bucket prototypes.

Finally, I would like recognize the motivation for all of my professional activities:

"The aeronautical and space activities of the United States shall be conducted so as to contribute ... to the expansion of human knowledge of phenomena in the atmosphere and space... The Administration shall ... provide for the widest practicable and appropriate dissemination of information concerning its activities and results thereof ..."

-- National Aeronautics and Space Act, 1958.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Overview and Objective

In aerospace engineering, Multidisciplinary Design Optimization (MDO) is a growing field concerned with the integrated design and analysis of applications using a combination of mathematical, engineering, and economic models and tools. Surpassing its early analysis and optimization roots, MDO now also includes "functions of interdisciplinary communication" [37]. To hasten the desired adoption of MDO methodology by other engineering research communities such as electrical and chemical engineering [37], NASA is acutely interested in the creation of a unified, canonical digital library of Scientific and Technical Information (STI).

The objective of our work was to study the feasibility of modifying existing digital library systems and protocols to support the multi-discipline and multi-format requirements. By using existing, popular digital library systems as the base for our work, we could more quickly design, develop, and test the advanced concepts necessary for multi-discipline and multi-format collections. Extending existing systems also increases the chances of acceptance and usage from both the researchers that use the digital library (DL) , and the organizations and individuals that publish into the DL.

1.2 Problem Statement

Spurred by recent advances in network information systems such as the World Wide Web (WWW), DLs are the topic of research in many scientific communities. However, 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 WWW-oriented digital library projects spanning over 5 different disciplines [8]. In short, each scientific community is hand crafting their own digital library infrastructure. Aside from the duplication of effort, there is a risk that each discipline will become less knowledgeable of other disciplines, as geographic balkanization gives way to "electronic balkanization in 'topic-space'" [41].

There are two significant problems with current digital libraries. First, multi-discipline research is difficult because the collective knowledge of each discipline is stored in incompatible DLs that are known only to the specialists in the field. In the MDO example above, current DL practices prohibit a structural engineer's awareness of computer or mathematical tools developed by another discipline that might be relevant to their research. The second significant problem with digital libraries is that although technical information consists of manuscripts, software, datasets, etc., the manuscript receives the majority of attention, and the other components are often discarded [39]. Past format restrictions have forced an artificial partitioning of the STI output along format lines (software tapes, photo negatives, printed reports, etc.). Although non-

manuscript digital libraries such as the software archive *Netlib* [4] are successful, they still placed the burden of STI reintegration on the customer. NASA customers desire to have the entire set of manuscripts, software, data, etc. available in one place [33]. With the increasing availability of all-digital storage and transmission, the re-integration of the STI output back to its original state is possible.

1.3 Approach

NASA Langley Research Center and Old Dominion University have established NCSTRL+ to research methods which address the multi-discipline and multi-format problems. NCSTRL+ is based on the Networked Computer Science Technical Report Library (NCSTRL) [6], which is a highly successful digital library offering access to over 80 different university departments and laboratories since 1994. It is implemented using the Dienst protocol [20]. NCSTRL+ initially includes selected holdings from the NASA Technical Report Server (NTRS) [26] 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 are published in *buckets* [28, 29], 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.).

A digital library containing buckets is important because it allows the mixing of traditional peer-reviewed literature ("white literature"), pre-prints and technical reports ("gray literature"), and data types and formats not generally archived well or at all (datasets, software). This is especially important to NASA Langley Research Center, where the demand and capability to produce test data has far outstripped the Center's ability to publish the summarized data through the traditional channels. Buckets will provide a tool for strategic data management at NASA, and multi-discipline clusters will accelerate the growth of emerging fields such as multi-disciplinary optimization.

The outline of the rest of this thesis is as follows: Chapter two provides an overview of digital libraries and some of the fundamental issues in this field. Chapter three provides an overview of the Dienst protocol and the modifications that were made to support clusters. Chapter four discusses buckets and other similar technologies. Chapter five discusses the NCSTRL+ testbed and the resulting observations. Chapter six introduces some areas for future work, including extending the cluster and bucket functionality as well as some additional services that clusters and buckets make possible. We conclude with Chapter seven.

CHAPTER TWO

OVERVIEW OF DIGITAL LIBRARIES

2.1 History of STI Exchange

Rapidity and breadth of communication have always been significant requirements in the exchange of STI. Scientific journals evolved in the 17th century to replace the system of exchanging personal letters between scientists, which evolved because of unacceptable delays in publishing books [31]. However, journals are no longer used for rapid communication, but rather as "a medium for priority claiming, quality control and archiving scientific work." [3] To achieve rapid communication of STI, different disciplines have adopted various models. In computer science, the technical report is a common unit of exchange. In some disciplines, such as high-energy physics, the pre-print culture is well established. Paul Ginsparg, a physicist active in digital libraries, notes that "The small amount of filtering provided by refereed journals plays no effective role in our research." [11] While noting that not all disciplines embrace the pre-print / technical report culture equally, Odlyzko [31] states "it is rare for experts in any mathematical subject to learn of a major new development in their area through a journal publication" and also relates comments by computer scientists Rob Pike ("that in his area journals have become irrelevant") and Joan Feigenbaum ("if it didn't happen at a conference, it didn't happen").

A journal article is often only a fraction of the available technical literature about a given subject. Theses, dissertations, conference papers, and technical reports are known as "gray literature" and receive varying degrees of peer review. "White literature," available through standard publications channels and processes, is often supported by a larger body of gray literature. The role of the large amount of gray literature and its relation to the smaller amount of white literature, and the issues associated with integrating the two have been present since the post-World War II U.S. Government sponsored research boom [2, 13, 36]. David Patterson, co-inventor of the RISC computer chip, noted that in one of his first research projects, the output was 2 journal articles, 12 conference papers, and 20 technical reports [32]. If we consider this pyramid of publications (Figure 1) to be typical, then a journal article actually functions as an abstract of a larger body of STI.

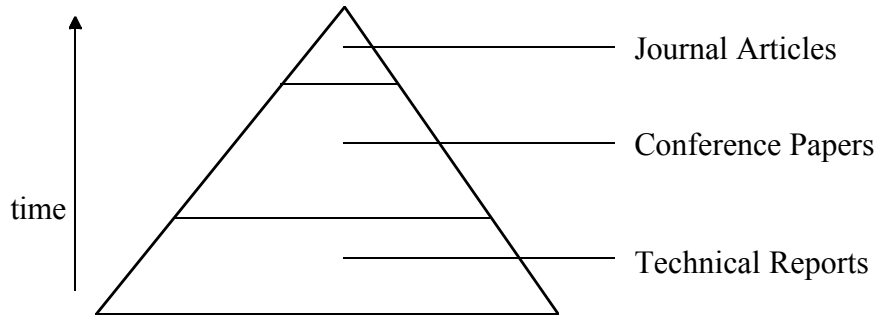


Figure 1. Pyramid of Publications for a Single Project/Concept

It is estimated that there are over 100,000 domestic technical reports produced annually [8]. The result is that even if there are 20,000 primary research journals [3], they do not represent the entirety of STI. These numbers do not include 1) confidential, secret, proprietary, and otherwise restricted reports; or 2) non-report STI, such as

computer software, data sets, video, geographic data, etc. Indeed, anecdotal evidence suggests that the WWW is not just a rapid transport mechanism for white and gray literature, but collections of WWW pages are becoming a new unit of STI exchange as well. Figure 2 shows the Pyramid of Publications resting on a larger body of unpublished STI.

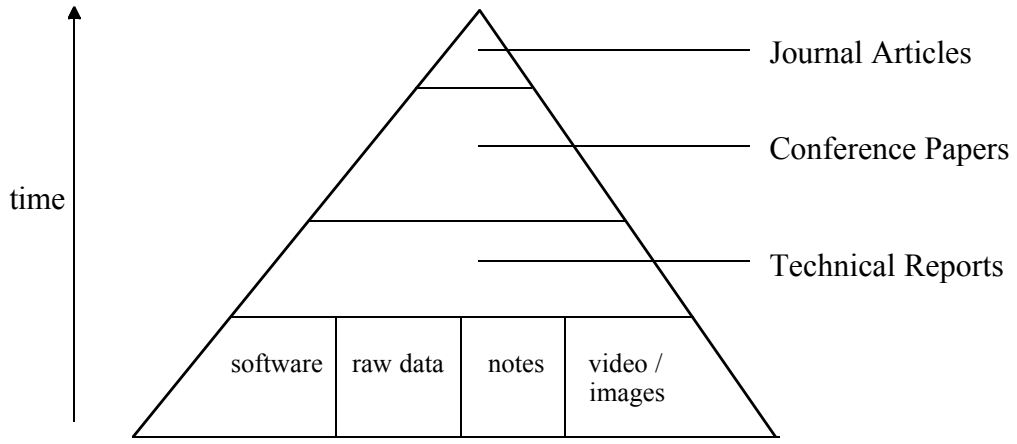


Figure 2. Pyramid of Publications Rests on Unpublished STI

Schatz and Chen [34] give a summary of current research projects focusing on building large digital libraries of non-report STI. However, these efforts can be summarized as propagating a "separate but equal" philosophy with regards to non-report STI. Instead of integrating software, datasets, etc. into the same DL which has the reports, separate DLs are created for the new collection. The researcher is still left to reconstruct the original information tuple by integrating search results from various DLs. The limitations of current STI exchange mechanisms can be summarized as:

- *highly focused on journal articles*, despite their decreasing value to researchers and practitioners in some fields;
- *inadequate acquisition of gray literature*, the grist of technical exchange;
- *inability to offer non-publication media*, such as datasets, software, and video.

These limitations are largely side effects of the hard copy distribution paradigm. As STI exchange moves toward electronic distribution, existing mechanisms should not merely be automated, but the entire process should be revisited.

2.2 Digital Library Models

The term "digital library" is broad and can mean different things in various contexts. Some projects, especially those aimed at preserving and converting historical collections into digital format, are obviously centered around library, archival, and preservation issues. However, if the focus includes current and future STI, then digital library projects are just as much about publishing and data management as libraries. Figure 3 illustrates how digital libraries cover both traditional library and publishing realms. Figure 3 also shows the projected impact area of NCSTRL+, specifically the present, future, and immediate past. Although Dienst has facilities to process scanned publications, its main function is to handle present and future publishing. It has applicability to the immediate past in that those documents are more likely to continue to exist in electronic format.

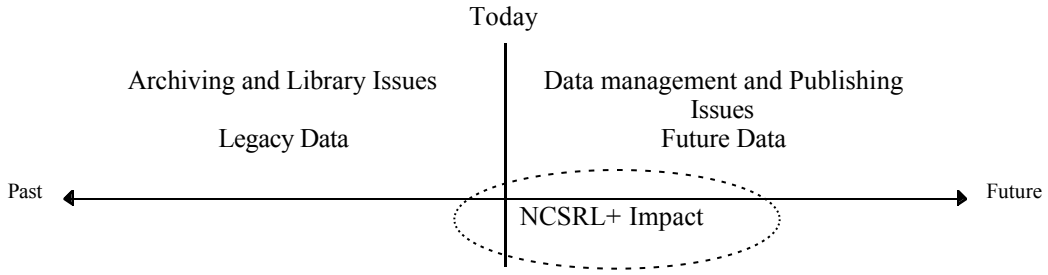


Figure 3. Digital Libraries Address Both Legacy and Future Data

From [8], we have a nomenclature for describing various DL projects. They can be differentiated by their architecture (distributed or centralized) and by the identity of the sponsor of the DL (traditional publishers or authoring individuals/groups). These classification factors are important because 1) distributed architecture DLs are more likely to be scalable then centralized DLs; and 2) a DL that accepts input from the authoring organization / individual is more likely to be able to capture the non-manuscript STI since the DL is "closer" to the information source. DLs that focus on traditional publication processes (i.e. journals), are less likely to provide access to non-manuscript STI since the DL is far removed from the information source. Figure 4 illustrates the partitioning along with the abbreviations for their taxonomy, and provides some example DLs of each type.

	distributed	centralized
traditional publisher	DLI ***DP***	CORE ***CP***
authoring individual / organization	NTRS, NCSTRL, NCSTRL+ ***DO***	WATERS ***CO***

Figure 4. Distribution of Digital Libraries

Centralized Architecture, Traditional Publisher (CP) - Input is from traditional publishing sources such as journals and professional societies, and all input is collected in a single physical and logical location. The server is either up or down, there is no graduated level of availability. An example of a CP DL is the Chemistry On-line Retrieval Experiment [7]. CORE converted several major chemistry journals into SGML, and provided access via a specialized X Window System client.

Distributed Architecture, Traditional Publisher (DP) - Input is from traditional publishing sources such as journals and professional societies, but the input is not transmitted to a single physical location. The user interface may give the appearance of a central location, but the service is comprised of several servers. An example of a DP DL is the University of Illinois' portion of the Digital Library Initiative (DLI) [35]. The DLI provides access to several major journals across multiple disciplines in a distributed architecture system.

Centralized Architecture, Authoring Individual/Organization (CO) - Input is from either individuals (a few papers at a time) or from an organization (papers transmitted in batches) and the input is transferred to a central location for indexing, processing and redistribution. An example of a CO DL is the Wide Area Technical Report Server (WATERS) [23], a project that eventually provided the foundation for NCSTRL Lite.

Distributed Architecture, Authoring Individual/Organization (DO) - Input could still be from individuals, but separate servers encourage clustering of publishers along organizational boundaries. Input stays at the server to which it was posted and the user interface handles querying all appropriate servers and collating and presenting the results. NTRS, NCSTRL, and NCSTRL+ are all examples of DO DLs.

2.3 Multi-Discipline Digital Libraries

There are three ways to build a multi-discipline digital library:

1. allow each discipline to build their own digital library protocol, and the multi-discipline library is constructed by providing a gateway function to the heterogeneous servers; or
2. propagate a given technology or protocol to many disciplines, and the multi-discipline library is constructed by the union of many homogenous servers; or
3. a combination of 1 and 2.

Lyceum [22] and STARTS [12] are examples of projects that focus on gatewaying and translating requests between heterogeneous servers. Lyceum provides access to heterogeneous servers by applying heuristics to change the search syntax and reformat the results based on the servers' protocols. STARTS actually defines a language for inter-server communication.

On the other hand, NCSTRL+ focuses on providing an extensible protocol that can be used by many disciplines. There will probably never be a single DL protocol in use by all sites, but rather a small set of several protocols in use. The eventual job of translating between these protocols will be made easier if a scalable and extensible DL

protocol is propagated before various disciplines have such a large investment in their own custom DL protocol that they cannot easily switch to a Dienst variant.

CHAPTER THREE

CLUSTERS OF DIENST SERVERS

3.1 History and Overview of Dienst

NCSTRL+ is the result of several years of research and development in digital libraries (Figure 5). In 1992, the ARPA-funded CS-TR project began [14] as did the Langley Technical Report Server (LTRS) [27]. In 1993, WATERS shared a code base with LTRS. In 1994, LTRS launched the NTRS, and the CS-TR and WATERS projects formed the basis for the current NCSTRL. In 1997, NTRS and NCSTRL formed the basis for NCSTRL+.

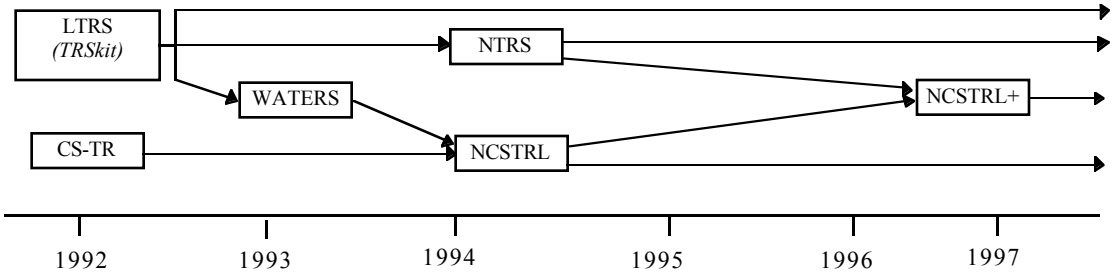


Figure 5. NCSTRL+ Lineage

We chose to implement NCSTRL+ using Dienst instead of other digital library protocols such as TRSkit [30] because of Dienst's success over several years of production in NCSTRL. Dienst appears to be the most scalable, flexible, and extensible of digital library systems we surveyed [8]. Dienst also serves as the basis for other digital library projects, including: the Electronic Thesis and Dissertation Project [9], the

University of Virginia undergraduate engineering thesis project [40] and the ACM SIGIR conference proceedings project (which requires ACM authentication) [1].

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://machine.name:port/Dienst/Service/Version/Verb/Arguments

An example of a valid Dienst request is:

http://repository.larc.nasa.gov:8080/Dienst/Meta/2.0/Publishers/

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 [15]. 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 [21].

The basic architecture of a Dienst system 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 the 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) and achieve scalability through this hierarchical sharing of information about where the various NCSTRL participants are on the network.

3.2 Clusters

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 (see Figure 14 for an example how the clusters are presented to the user). NCSTRL already has a single default cluster of publishing authority, which in practice generally maps to the author's organization. *Genre* is a term provided by Ed Fox in a private communication and refers to distinguishing between journal articles, technical reports, theses and dissertations, etc. Subject category is the cluster that provides the ability for multi-disciplinary collections to be developed. Although servers for different disciplines could be integrated into NCSTRL+, there would be no provision for viewing

or searching subject based subsets of the entire collection without the subject category cluster.

Dienst currently carries no concept of subject category or genre in its protocol, despite having provisions for specifying keywords from the title, author(s), and abstract. In fact, digital libraries using the Dienst protocol that register with NCSTRL have the implicit assumption that all holdings are computer science related. To demonstrate the addition of subject category and genre to Dienst, we added new fields for some of the message verbs (Table 1). Table 2 lists all the fields supported in NCSTRL+. The *authority* field is already defined in Dienst and serves as a default cluster.

Table 1. Modified Dienst Verbs

Service	Message Verb	New Argument	Argument Type
Index	SearchBoolean	ncstrlplus_sti_topics ncstrlplus_search_genres	optional
User Interface	QueryNF	implicitly applies to all fields	N/A
User Interface	Search	none; default output modified to include the new fields	N/A

Table 2. Defined Fields in NCSTRL+

Field	New to NCSTRL+
ncstrlplus_sti_topics	yes
ncstrlplus_search_genres	yes
authority	no
author	no
title	no
abstract	no

The verb modifications listed in Table 1 were our initial attempt to demonstrate cluster functionality to the user. The term "clusters" for this purpose is due to Carl 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. Table 3 shows the files from the Dienst 4.1.8 distribution that were modified to produce the initial functionality. Minimizing the number of source code modifications necessary was a high priority.

Table 3. Files Modified in Dienst 4.1.8 Distribution

Modified File	Comments
Config/install.config	modification required for Dienst installation
Config/config_constants.pl	cosmetic changes in HTML output
UI_Server/search.pl	changes in printing the search interface
UI_Server/trs.pl	bucket invocation
Meta/data.pl	missing file needed for networked NCSTRL+ operation
Indexer/search_db.pl	searching for new fields
Indexer/parse_bib_file.pl	reading in new fields from bib file
Indexer/build-inverted-indexs.pl	indexing new fields

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 4 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 4. 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
User Interface	Search	none; would modify default output to include cluster selector	N/A
User Interface	QueryNF	cluster=	optional
User Interface	BrowseYears	cluster=	optional
User Interface	ListYears	cluster=	optional
User Interface	BrowseAuthors	cluster=	optional
User Interface	ListAuthors	cluster=	optional
Library Management	ListClusters (proposed)	none	none
Library Management	DescribeClusters (proposed)	none	none

For the NCSTRL+ prototype, we adopted the NASA STI subject categories. A full listing of the subject categories can be found in [24]. 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 5). 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 5. 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 (Figure 6). The NASA STI topics for NCSTRL+ will be implemented as a new optional and repeatable field in RFC-1807 format.

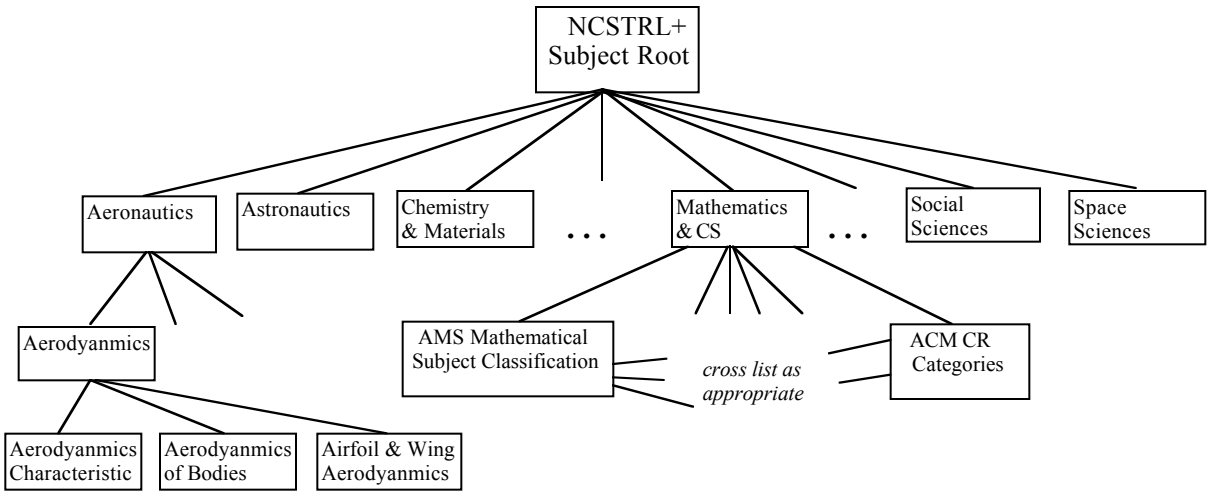


Figure 6. NCSTRL+ Subject Tree

Table 6 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 6. 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

CHAPTER FOUR

BUCKETS

4.1 Bucket Definition

Buckets are entities for publishing and archiving in digital libraries. 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 a CD-ROM, but these items are likely to 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. Figure 7 shows the stored and lost portions of STI in the project / archival / reuse process.

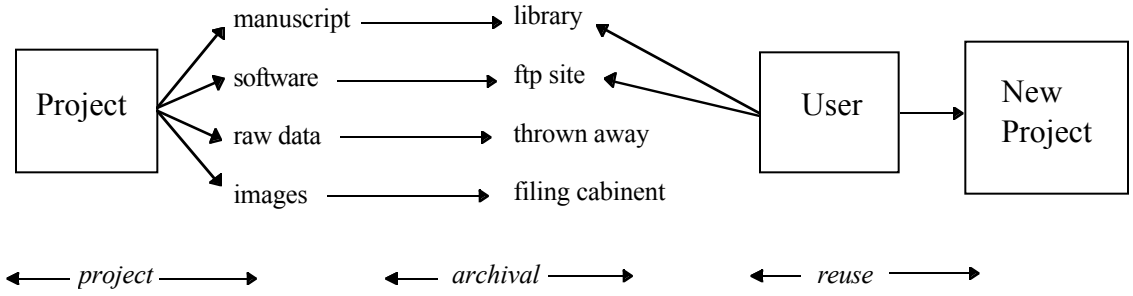


Figure 7. STI Lost in Project / Archival / Reuse Process

Buckets solve this problem by allowing all the disparate information media types to be integrated digitally in a single object. Buckets also provide a mechanism to address the increasing problem of loss of unpublished data. Currently, Langley produces significant amounts of data that is passed directly to its customers and a NASA report is never written to document the project. Sometimes this is because information is lost before a summary can be written, and with another project on the horizon, there is little time to devote to documenting a project for future research use. Bucket technology will allow a mechanism to collect unpublished data into a logical collection, either for current use or to allow future documentation and publication about the project.

Buckets are designed to be highly customizable and unique. It would be possible for large archives to have no buckets with exactly the same functionality. 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 (PostScript, 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 [39] by providing a common mechanism to keep related STI products together. A single bucket can have multiple *packages*. Packages can correspond to the semantics of the information (manuscript, software, etc.), or can be an abstract entity such as the metadata for the entire bucket, bucket terms and conditions, pointers to other buckets or packages, etc. A single package can have several *elements*, which are typically different file formats of the same information, such as the manuscript package having both PostScript and Adobe Acrobat (PDF) elements. Figure 8 illustrates the architecture of a typical bucket.

Buckets are similar in concept to the Digital Objects (DOs) of the Kahn-Wilensky Framework [16] and its derivatives. The NACA Report Server [25] is a digital library with bucket-like objects. Buckets are also somewhat similar to the various scientific data formats available such as netCDF, HDF, etc. [38], in that these formats seek to canonicalize and package disparate data representations. Buckets are also somewhat similar to experimental filesystems such as ELFS [17], which attempt to redefine the concept of "file" to include logical collections. The major distinguishing factor of buckets over other proposals is that buckets contain intelligence and can be active, where most other digital library objects, data formats, or filesystem objects are passive and the repository or other interface contains all the intelligence. Buckets combine elements of the above projects with the capability of intelligent agents.

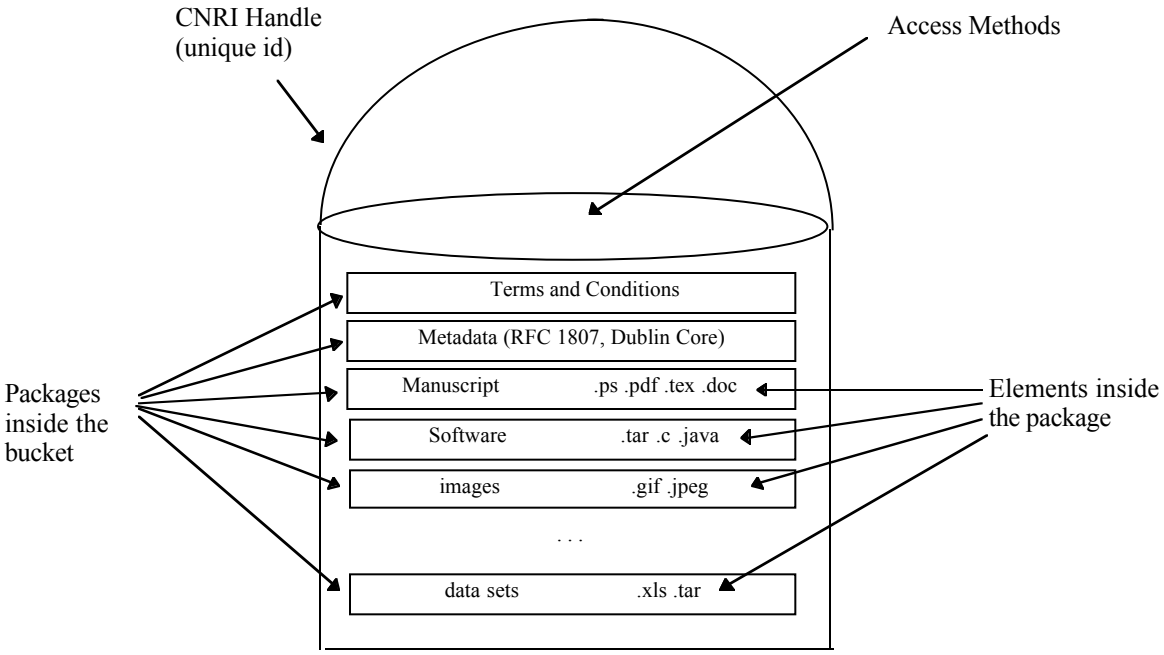


Figure 8. A Typical Bucket Architecture

4.2 Bucket Requirements

Buckets are intended to be either standalone objects or to be placed in digital libraries. They have unique ids (CNRI handles) associated with them. 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 (10s - 100s of thousands) of intelligent, custom buckets instead of large numbers (millions) of homogenous buckets. Figure 9 contrasts a traditional architecture of having the repository interface contain all the intelligence and functionality with that of a bucket architecture where the repository intelligence and functionality can be split between the repository and individual buckets. This could be most useful when individual buckets require custom terms and conditions

for access (security, payment, etc.). Figure 9 also illustrates a bucket gaining some repository intelligence as it is extracted from the archive en route to becoming a standalone bucket. Table 7 lists some additional high level bucket requirements, with some terms such as "pointer" left undefined. Table 8 lists the required bucket methods; other methods can be custom defined. Note that Table 8 differs from protocols such as the Repository Access Protocol (RAP) [19] 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 7. Bucket Requirements

Number	Requirement
1.	a bucket is of arbitrary size
2.	a bucket has a globally unique identifier
3.	a bucket contains 0 or more components, called packages (no defined limit)
4.	a package contains 1 or more components, called elements (no defined limit)
5.	an element can be a file, or a "pointer" to another
6.	both packages and elements can be other buckets (i.e., buckets can be nested)
7.	a package can be a "pointer" to a remote bucket, package, or element (remote package or element access requires "going through" the remote hosting bucket)
8.	buckets can keep internal logs of actions performed on them
9.	interactions with packages or elements are made only through defined methods on a bucket
9.	buckets can initiate actions; they do not have to wait to be acted on
10.	buckets can exist inside or out of a repository

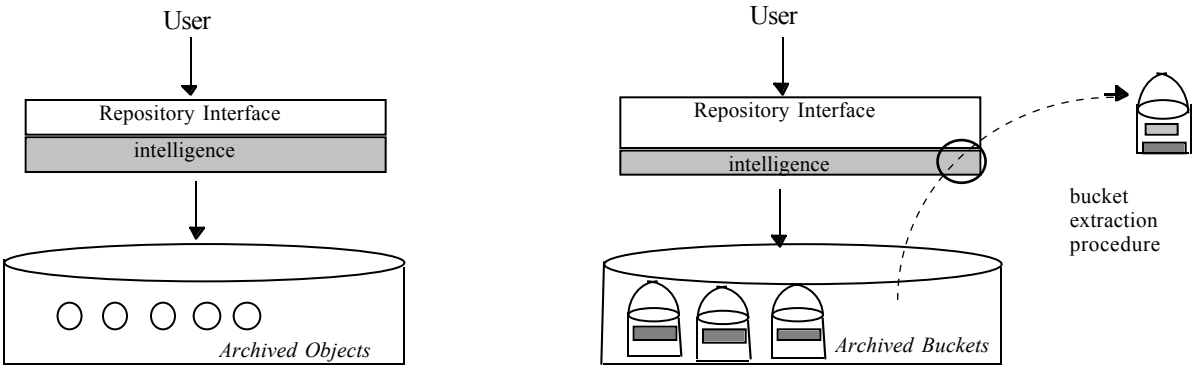


Figure 9. Traditional and Bucket Repository Architectures

Table 8. 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)
terms_and_conditions	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
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

4.3 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 code is saved in an index.cgi file, which is a WWW method for defining a Common Gateway Interface (CGI) script to be run if no other file name for the directory is given. Thus, any attempt to access a directory forces the invocation of the index.cgi file, thus protecting the bucket contents from browsing access. The index.cgi file has the responsibility for regulating access to the bucket contents, insuring that terms and conditions are met, negotiating presentation strategy with the user's WWW client, etc.

The philosophy behind 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, the Dienst protocol has 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. This allows for a bucket to redefine its default method to be terms_and_conditions, so the same repository can have some buckets secured, and others could be open to all.

Our method of shifting responsibility from Dienst to the bucket does not build in an explicit HTTP dependency, since the local User Interface service is not accessing the index.cgi script across the network, but rather simply executing a script that resides on its local filesystem. The fact that the script produces HTML output is not important to Dienst since the bucket is now handling all future interactions. The shift in display responsibility is shown in Figure 10.

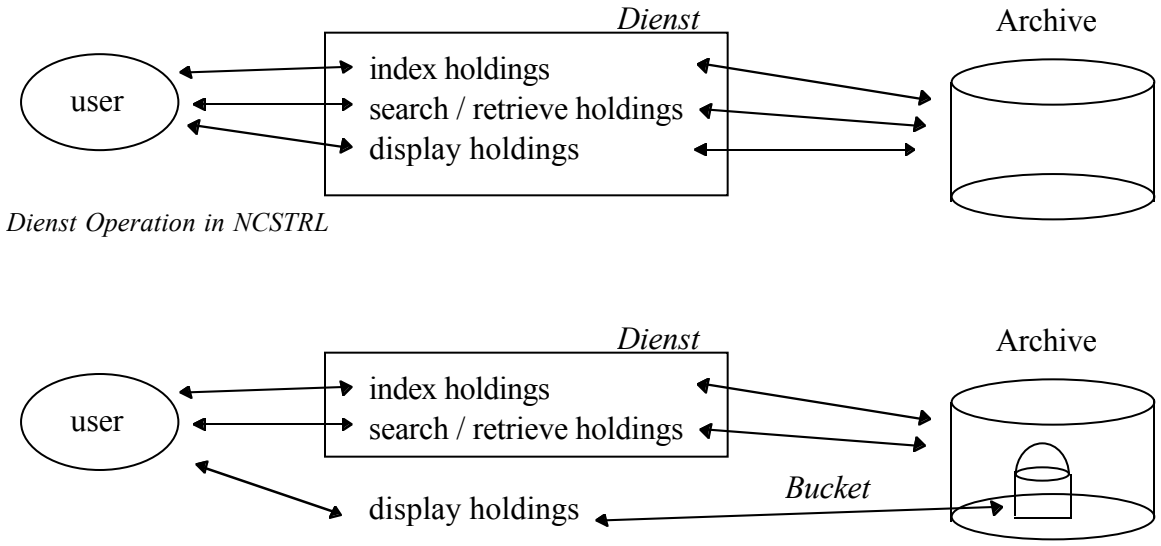


Figure 10. Buckets, Not Dienst, Handle Display in NCSTRL+

CHAPTER FIVE

NSCTRL+ TESTBED

5.1 Architecture

Figure 11 shows the architecture of NCSTRL+. Three machines are employed. The first will be the home page and meta data collection/search machine, and will reside at NASA. NASA will also house a second machine for the aeronautics cluster. Old Dominion will use a third machine to host the computer science cluster. Although similar in appearance, the NCSTRL+ prototype will be operationally independent of NCSTRL. At present, only a single machine is employed. To create a networked collection independent of NCSTRL requires a file (Meta/data.pl) that is not included in the standard Dienst distribution. The Dienst authors have been contacted and when this file is obtained, the architecture in Figure 11 will be achieved.

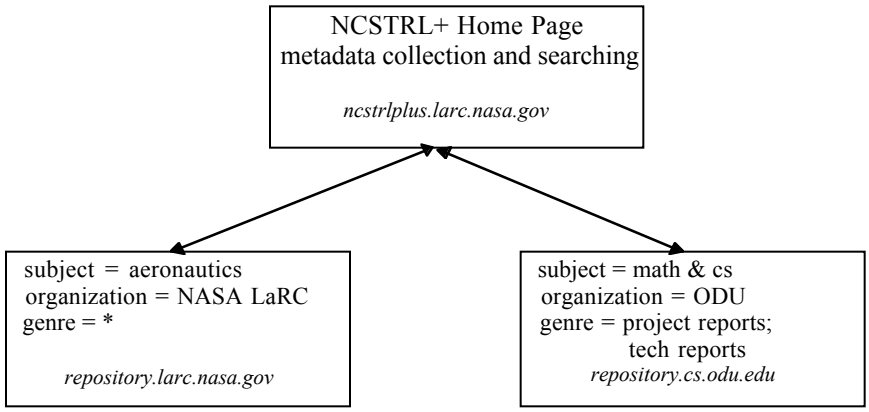


Figure 11. Initial NCSTRL+ Architecture

5.2 Metadata

Currently, all NCSTRL+ buckets use the RFC-1807 metadata format. However, any format can be used and Dublin Core [18] is a likely format to be adopted in the future. There is no reason that multiple metadata formats cannot be simultaneously supported. Although logically, a bucket has its own metadata, and all its packages have their own separate metadata, the implementation is such that all the package metadata fields can be embedded with the single metadata file for the bucket. It is this single metadata file that is indexed. This allows the package metadata to be searched simultaneously, and the linkage is created so that multiple hits across many packages within a single bucket will produce only one bucket to be returned. Figure 12 shows a sample metadata file from NCSTRL+.

5.3 Searching NCSTRL+

NCSTRL+ searching is similar to searching with NCSTRL, with the addition of specifying desired clusters to search. Figure 13 shows the NCSTRL+ home page, which is designed to look as much like NCSTRL as possible. The default search from the NCSTRL+ home page applies to all clusters. Figure 14 shows how the advanced fielded search form of NCSTRL+ is modified, allowing the selection of desired subject categories and data genres. Figure 15 shows a sample search results page, including 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 similar to the choices available to current users of NCSTRL.

BIB-VERSION:: X-NCSTRL+1.0
ID:: ncstrlplus.larc//SAE-97ES-29
ENTRY:: July 16, 1997
ORGANIZATION:: NASA Langley Research Center
TYPE:: 05
TITLE:: Thermal Design and Analysis for the Cryogenic MIDAS Experiment
AUTHOR:: Amundsen, Ruth M.
DATE:: July 1997
PAGES:: 9
HANDLE:: hdl:ncstrlplus.larc//SAE-97ES-29
OTHER_ACCESS::
KEYWORD:: Cryogenic
KEYWORD:: Thermal analysis
KEYWORD:: Spaceflight
KEYWORD:: High temperature superconductors
CR-CATEGORY::
NCSTRL+CATEGORY:: 76
COPYRIGHT:: U.S. Government works are not copyrighted.
ABSTRACT:: The Materials In Devices As Superconductors (MIDAS) spaceflight experiment is a NASA payload which launched in September 1996 on the Shuttle, and was transferred to the Mir Space Station for several months of operation. MIDAS was developed and built at NASA Langley Research Center (LaRC). The primary objective of the experiment was to determine the effects of microgravity and spaceflight on the electrical properties of high-temperature superconductive (HTS) materials. The thermal challenge on MIDAS was to maintain the superconductive specimens at or below 80 K for the entire operation of the experiment, including all ground testing and 90 days of spaceflight operation. Cooling was provided by a small tactical cryocooler. The superconductive specimens and the coldfinger of the cryocooler were mounted in a vacuum chamber, with vacuum levels maintained by an ion pump. The entire experiment was mounted for operation in a stowage locker inside Mir, with the only heat dissipation capability provided by a cooling fan exhausting to the habitable compartment. The thermal environment on Mir can potentially vary over the range 5 to 40 Degrees C; this was the range used in testing, and this wide range adds to the difficulty in managing the power dissipated from the experiment's active components. Many issues in the thermal design are discussed, including: thermal isolation methods for the cryogenic samples; design for cooling to cryogenic temperatures; cryogenic epoxy bonds; management of ambient temperature components' self-heating; and fan cooling of the enclosed locker. Results of the design are also considered, including the thermal gradients across the HTS samples and cryogenic thermal strap, electronics and thermal sensor cryogenic performance, and differences between ground and flight performance. Modeling was performed in both SINDA-85 and MSC/PATRAN (with direct geometry import from the CAD design tool Pro/Engineer). Advantages of both types of models are discussed. Correlation of several models to ground testing and flight data (where available) is presented. Both SINDA and PATRAN models predicted the actual therma performance of the experiment well, even without post-flight correlation adjustments of the models.
PACKAGE:: ncstrlplus.larc//SAE-97ES-29/P1
PACKAGE-TITLE:: Conference Manuscript
PACKAGE-ABSTRACT:: Presented at 27th International Conference on Environmental Systems, Lake Tahoe, Nevada, July 14-17, 1997.
ELEMENT-FILE:: ncstrlplus.larc//SAE-97ES-29/P1/97ES-29.doc
ELEMENT-APP:: MS Word 97 document
ELEMENT-FILE:: ncstrlplus.larc//SAE-97ES-29/P1/NASA-97-27ices.ps.Z
ELEMENT-FILE:: ncstrlplus.larc//SAE-97ES-29/P1/NASA-97-27ices.pdf
PACKAGE-END:: ncstrlplus.larc//SAE-97ES-29/P1
PACKAGE:: ncstrlplus.larc//SAE-97ES-29/P2
PACKAGE-TITLE:: Conference Presentation
PACKAGE-ABSTRACT:: Presented at 27th International Conference on Environmental Systems , Lake Tahoe, Nevada , July 14-17, 1997.
ELEMENT-FILE:: ncstrlplus.larc//SAE-97ES-29/P2/SAE_pres97.ppt
ELEMENT-APP:: MS PowerPoint 97 presentation
PACKAGE-END:: ncstrlplus.larc//SAE-97ES-29/P2
END:: ncstrlplus.larc//SAE-97ES-29

Figure 12. Bucket + Package Metadata in a Single File (new fields in bold)

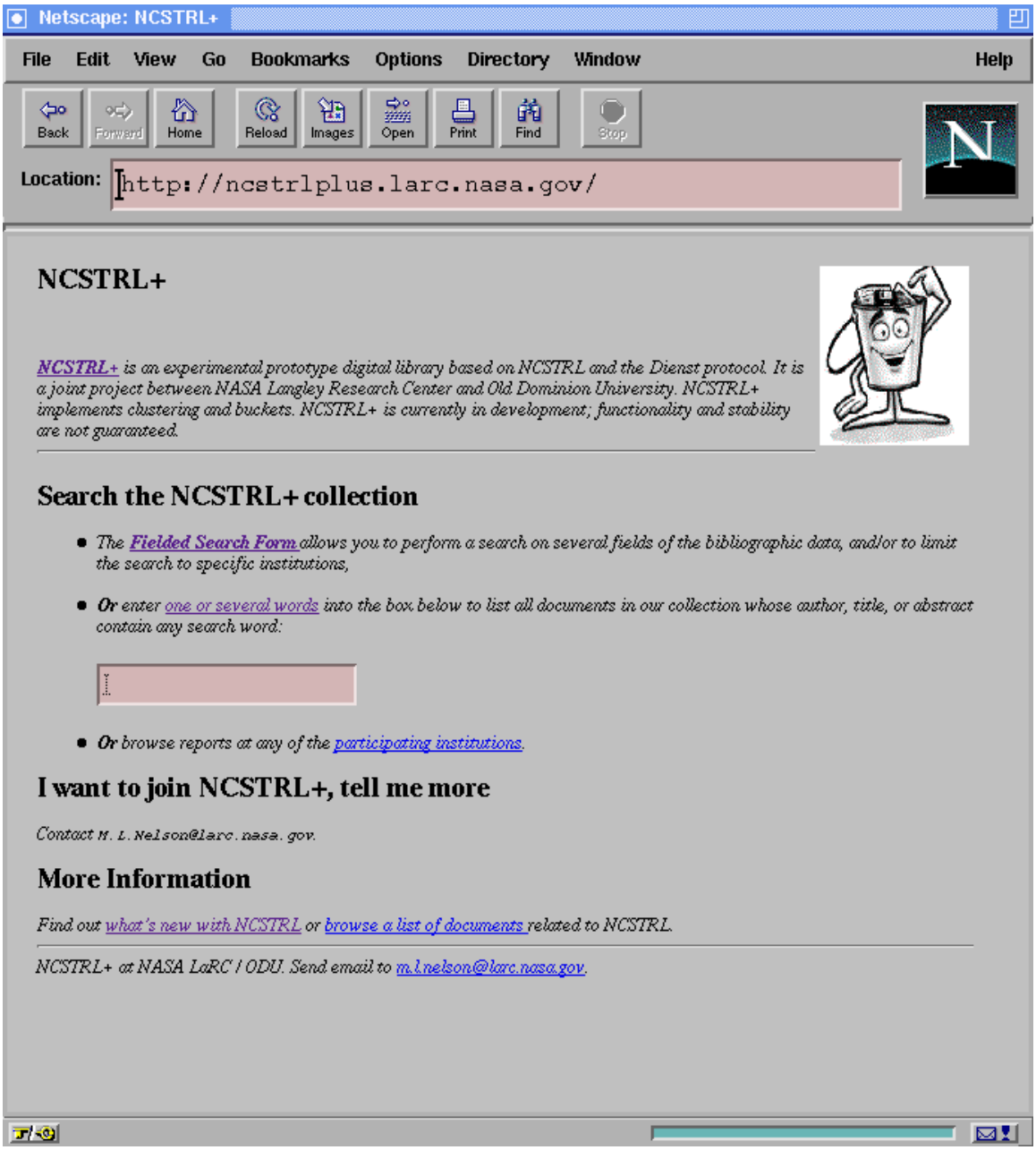


Figure 13. NCSTRL+ Home Page

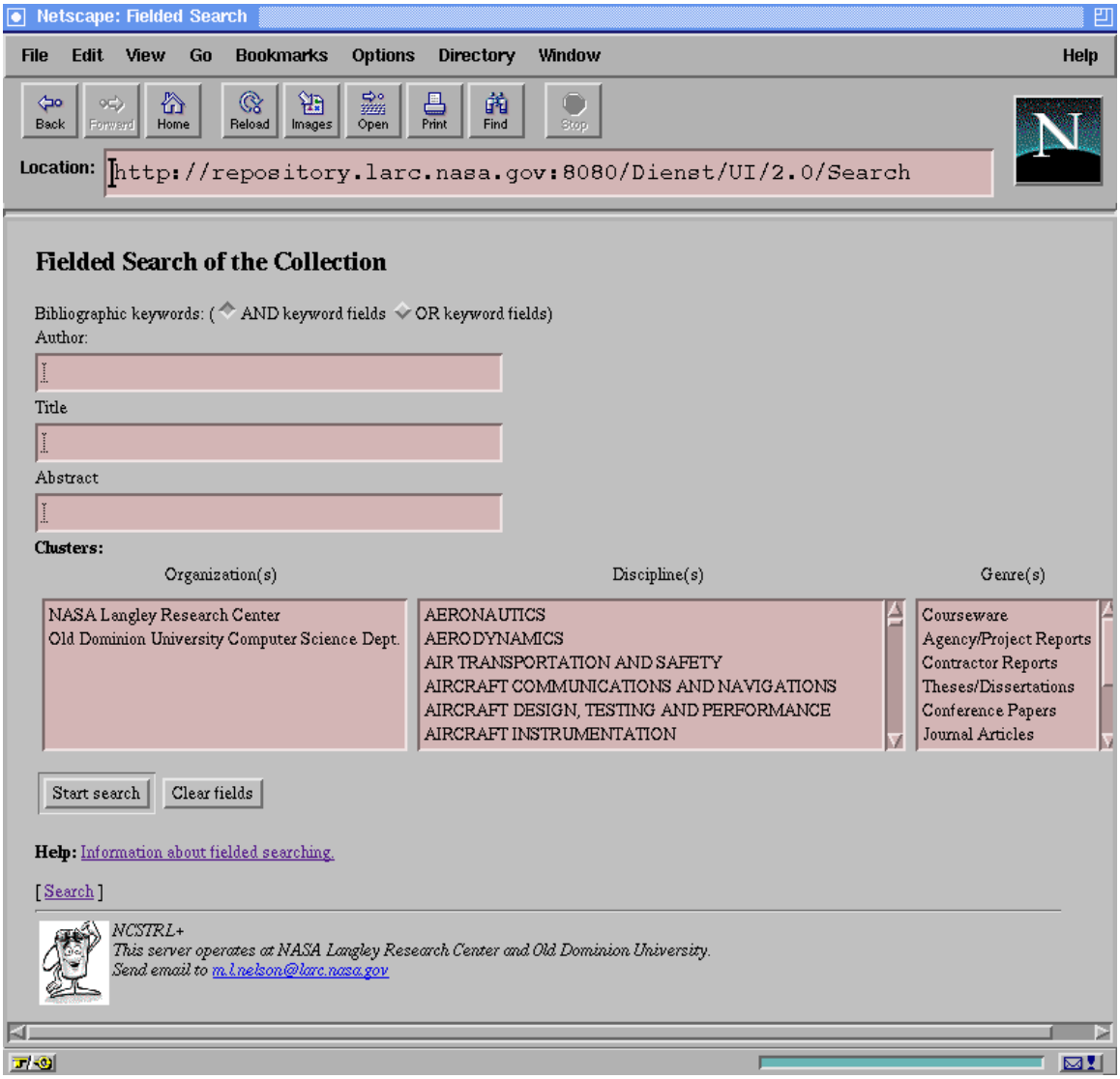
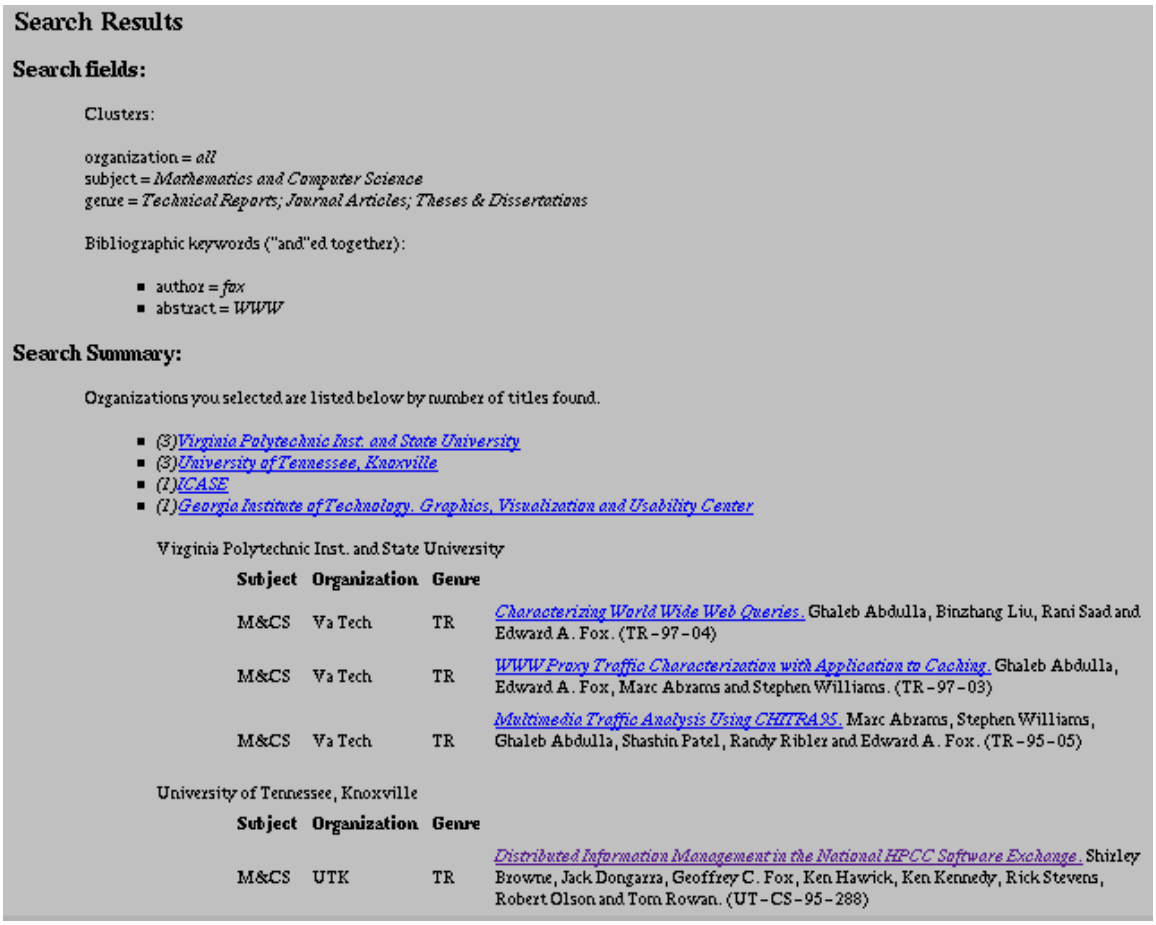


Figure 14. The Fielded Search Screen of NCSTRL+

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 available if the bucket has non-manuscript packages. Figure 16 illustrates a typical bucket with packages other than a manuscript. The interface is similar to NCSTRL, with the exception that the additional data semantics are presented (software, datasets, etc.).



Search Results

Search fields:

Clusters:

organization = *all*
subject = *Mathematics and Computer Science*
genre = *Technical Reports; Journal Articles; Theses & Dissertations*

Bibliographic keywords ("and"ed together):

- author = *fox*
- abstract = *WWW*

Search Summary:

Organizations you selected are listed below by number of titles found.

- *(3)Virginia Polytechnic Inst. and State University*
- *(3)University of Tennessee, Knoxville*
- *(1)ICASE*
- *(1)Georgia Institute of Technology. Graphics, Visualization and Usability Center*

Virginia Polytechnic Inst. and State University

Subject	Organization	Genre	
M&CS	Va Tech	TR	*Characterizing World Wide Web Queries.* Ghaleb Abdulla, Binzhang Liu, Rani Saad and Edward A. Fox. (TR-97-04)
M&CS	Va Tech	TR	*WWW Proxy Traffic Characterization with Application to Caching.* Ghaleb Abdulla, Edward A. Fox, Marc Abrams and Stephen Williams. (TR-97-03)
M&CS	Va Tech	TR	*Multimedia Traffic Analysis Using CHITRA95.* Marc Abrams, Stephen Williams, Ghaleb Abdulla, Shashin Patel, Randy Ribler and Edward A. Fox. (TR-95-05)

University of Tennessee, Knoxville

Subject	Organization	Genre	
M&CS	UTK	TR	*Distributed Information Management in the National HPCC Software Exchange.* Shirley Browne, Jack Dongarra, Geoffrey C. Fox, Ken Hawick, Ken Kennedy, Rick Stevens, Robert Olson and Tom Rowan. (UT-CS-95-288)

Figure 15. NCSTRL+ Search Results Screen

5.4 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. At present, we are constructing buckets by hand so that we may continue to increase our understanding of what is required in a bucket. However, when the entire process is automated, publishing a manuscript in NCSTRL will be 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.

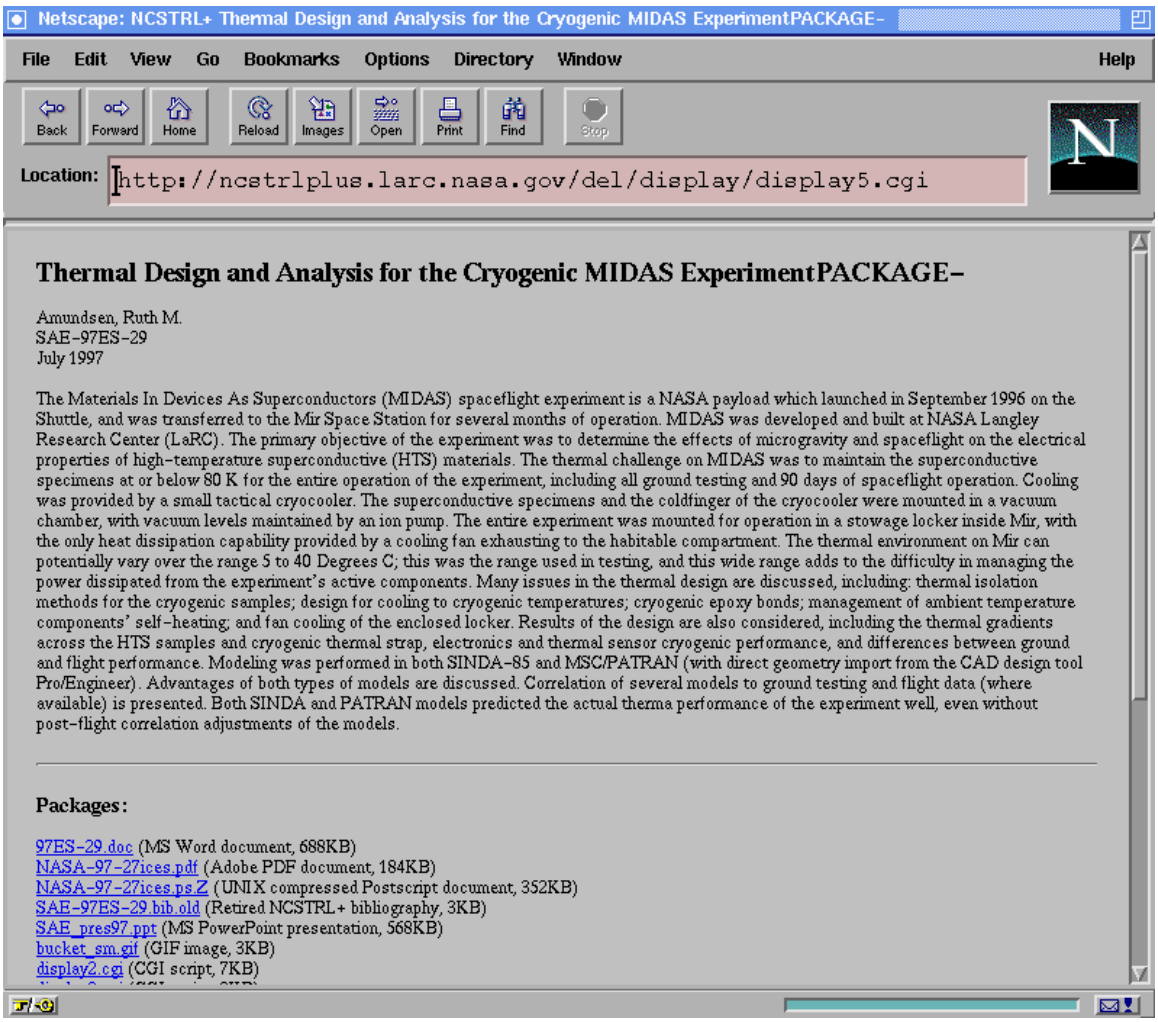


Figure 16. A Typical Bucket Presentation

CHAPTER SIX

FUTURE WORK

6. 1 Buckets and Bucket Tools

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. When we have created enough buckets by hand, we will develop tools to facilitate the creation and management of buckets. There are two main tools planned for bucket use. One is the *author tool*, which allows the author to construct a bucket with no programming knowledge. Figure 17 shows the prototype for the author tool. 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 (which lead to the cluster options available as shown in Figure 14). 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. Figure 18 shows the prototype for the management tool interface. 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.

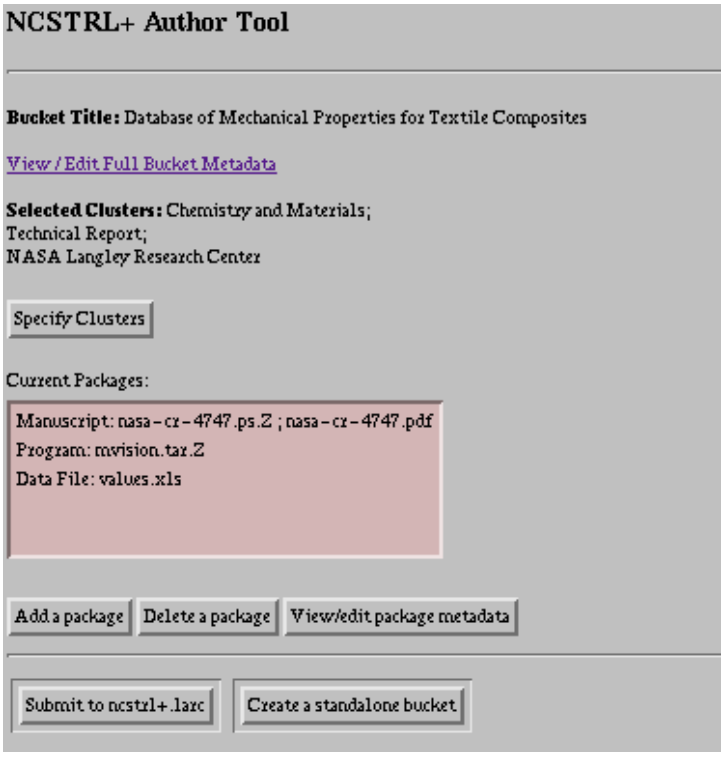


Figure 17. Author Tool

Figure 18. Management Tool

6.2 Bucket Matching System

The premise of the Bucket Matching System (BMS) is that the archived objects (buckets) should handle as many tasks as possible, not humans. Toward this end, we propose the BMS as a communications mechanism for buckets to exchange information among themselves. The "tuple-space" communication model of the Linda programming language [5] is the inspiration we draw upon. The following discusses some of the functionality but not the implementation.

The following example illustrates a usage of the BMS. Consider a technical report published by the CS department which is also submitted to a conference. The report appears under the server maintained by the department and publishing authority which is: ncstrl.odu.cs. If the conference paper is accepted, it will eventually be published by the conference sponsor, say ACM. The publishing authority would be ncstrl.acm. Although the conference paper will surely appear in a modified format, the tech report and the

conference paper are clearly related, despite being separated by publishing authority, date of publication, and revisions. Two separate but related objects now exist, and are likely to continue to exist. How best to create the desired linkage between the objects? “ncstrl.acm” has neither the resources nor the interest to spend the time searching out previous versions of a manuscript. “ncstrl.odu.cs” can not link to the conference bucket at the creation time of the ODU bucket, since the conference bucket did not exist at the time. It is unrealistic to suggest that the relevant parties will go back to the ncstrl.odu.cs collection and create the linkage correctly after 6 months have passed.

The solution is to have both buckets publish their metadata, or some subset of it, in the BMS. When a match, or near match, is found, the buckets can either 1) automatically link to each other; or more likely 2) bring the possible linkage to the attention of a person, who will provide the final approval for the linkage. There are a number of ways that a "match" can be found, but most likely it will be similar metadata within some definable threshold (e.g., 90% similar). Other uses for the BMS could include:

- *Find similar works by different authors.* The exact values would have to be determined by experimentation, but it possible to envision a similarity ranking that is slightly lower being an indication of a similar work by different authors. For example, a similar work by a different author would be:

  70% < similarity < 90%.

- *Arbitrary selective dissemination of information (SDI) services*. When a user's profile is matched, a notification can be sent immediately or a digest sent at every defined time interval (i.e., weekly). This method can be used to track different versions of a report, not just inter-genre (technical report vs. conference paper) or inter-institution (the author moves to a different university) issues. If version 2.0 of a bucket comes out, it can "find" all previous versions, and the appropriate actions can be taken (i.e., create a fully connected graph between the buckets, delete previous buckets, etc.)

- *Metadata scrubbing*. The issues of maintaining consistency and quality of meta data information is an increasingly important concern in digital libraries [10]. If part of the BMS also included a meta data scrubber that went through and based on rules and heuristics defined at the scrubber, it could automatically make or suggest updates to meta data. For example, having all references to "Hampton Institute" indicate the name change to "Hampton University", or handle an authors name change (such as if someone changes their name upon marriage), correct errors that may have been introduced, etc.

The BMS could be implemented on multiple workstations, and would be primarily batch processing. Given that some of the operations would be computationally expensive, it can be done with loose time guarantees, perhaps even done on stolen idle cycles (from "hallway clusters").

6.3 Hierarchical Clusters

An aspect of clustering we are eager to explore is the concept of hierarchical clusters. For example, under the current operation, ncstrlplus.odu.cs and ncstrlplus.odu.me would be registered as separate publishing authorities, and would not be subordinate to the handle ncstrlplus.odu. We feel that by delegating the naming authority for ncstrplus.odu.cs to ncstrlplus.odu, a more scalable approach to naming and searching is developed. The hierarchy functionality would apply to all clusters, not just publishing authorities. This would allow for greater economy in representing subject categories that are far from a given user's field. As the contents of NCSTRL+ grows, there will need to be an easy way to "zoom-in" and "zoom-out" for interesting clusters.

6.4 Additional Capabilities

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 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.

Since genre selection is not the purview of any profession, experience in choosing the right genre types will be gained with increased interaction by participants from other disciplines.

CHAPTER SEVEN

SUMMARY AND CONCLUSIONS

Due to the increased requirements for multidisciplinary activities, NASA is interested in the availability of a canonical, unified digital library for STI to counter the current trend of disciplines developing their own incompatible digital libraries. To achieve this goal of a multi-discipline, multi-format digital library, we have modified Dienst to study the feasibility of such a library. Dienst was chosen because 1) it has been in use in NCSTRL for several years; 2) it is an open, freely available architecture; 3) its distributed architecture allows for scalability; and 4) it maps well to the model of an organization (research center, laboratory, university department) as publisher. We have shown that few code modifications are needed to add the ability to cluster around subject category or information genre. There are many ways to provide such a clustering service; we have chosen to demonstrate its functionality by adding subject categories and genres as additional search fields. Since our modifications are limited in scope, we have noticed no change in the performance profile of NCSTRL+ versus NCSTRL. We have also designed and begun the initial prototype of buckets as a way to collect multiple data semantics and formats together. The NCSTRL+ prototype proves that it is possible and desirable to have a single digital library provide access to manuscripts, software, datasets, etc., as

opposed to the current trend of creating a separate digital library for each semantic or format type.

NCSTRL+ is forged from the holdings of the NCSTRL and NTRS archives and provides access to aerospace, mathematics, computer science, physics and engineering STI. These extensions are to gain user feedback on the usefulness of this service while awaiting the development of a generalized clustering service for Dienst. 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. Buckets also provide a method for capturing non-manuscript STI such as software and datasets that are currently poorly archived or not archived at all.

Although the associated social and political problems of changing the nature of an institution's publication vector are not addressed, NCSTRL+ provides a platform for experimentation for testing user response to multi-discipline clusters and logical collections of STI. At this point, we have no data concerning the usefulness of buckets and clusters to the user, or about their cost effectiveness. However, 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+. Contact information, current status, and related software can be found at: http://ncstrlplus.larc.nasa.gov/

References

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

2. J. Bennington, 'The Integration of Report Literature and Journals', *American Documentation*, 3(3), 149-152, (1952).

3. B. C. Bennion, 'Why the Science Journal Crisis?', *ASIS Bulletin*, February/March, 25-26, (1994).

4. S. Browne, J. Dongarra, E. Grosse, S. Green, K. Moore, T. Rowan, and R. Wade, 'Netlib Services and Resources', University of Tennesse Technical Report UT-CS-93-222, 1993.

5. N. Carriero and D. Gelernter, 'Linda in Context', *Communications of the ACM*, 32(4), 444-458, (1989).

6. J. R. Davis, D. B. Krafft, and C. Lagoze, 'Dienst: Building a Production Technical Report Server', *Advances in Digital Libraries*, Springer-Verlag, 1995, pp. 211-222.

7. R. Entlich, L. Garson, M. Lesk, L. Normore, J. Olsen, and S. Weibel, 'Making of a Digital Library: The Chemistry Online Retrieval Experiment', *Communications of the ACM*, 38(4), 54, (1995).

8. S. L. Esler and M. L. Nelson, 'The Evolution of Scientific and Technical Information Distribution', *Journal of the American Society of Information Science,* (In Press).

9. E. Fox, J. Eaton, G. McMillan, N. Kipp, L. Weiss, E. Arce, and 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*, September 1996. http://www.dlib.org/dlib/september96/theses/09fox.html

10. J. C. French, A. Powell and E. Schulman , 'Automating the Construction of Authority Files in Digital Libraries: A Case Study', University of Virginia Technical Report CS-97-02, January 1997.

11. P. Ginsparg, 'First Steps Towards Electronic Research Communication', *Computer in Physics*, 8, 333-341, (1994).

12. L. Gravano, C.-C. K. Chang, H. Garcia-Molina, A. Paepcke, 'STARTS: Stanford Proposal for Internet Meta-Searching', *Proceedings of the 1997 ACM SIGMOD International Conference On Management of Data*, 1997.

13. D. E. Gray, 'Organizing and Servicing Unpublished Reports', *American Documentation*, 4(3), 103-115, (1953).

14. R. Kahn, 'An Introduction to the CS-TR Project', December 1995. http://www.cnri.reston.va.us/home/describe.html

15. R. Kahn, 'The Handle System Version 3.0: An Overview', http://www.handle.net/docs/overview.html

16. R. Kahn and 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

17. 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.

18. C. Lagoze, C. A. Lynch, and R. Daniel, 'The Warwick Framework: A Container Architecture for Aggregating Sets of Metadata', Cornell University Computer Science Technical Report TR-96-1593, June 1996.

19. C. Lagoze and D. Ely, 'Implementation Issues in an Open Architectural Framework for Digital Object Services', Cornell University Computer Science Technical Report, TR95-1540, June 1995.

20. C. Lagoze, E.Shaw, J. R. Davis, and D. B. Krafft, 'Dienst: Implementation Reference Manual', Cornell Computer Science Technical Report TR95-1514, 1995.

21. R. Lasher, and D. Cohen, 'A Format for Bibliographic Records', Internet RFC-1807, June 1995.

22. M.-H. Maa, S. L. Esler and M. L. Nelson, 'Lyceum: A Multi-Protocol Digital Library Gateway', NASA TM-112871, July 1997.

23. K. Maly, J. French, E. Fox, and A. Selman , ''Wide Area Technical Report Service - Technical Reports Online,'' *Communications of the ACM*, 38(4), 45, (1995).

24. NASA Scientific and Technical Information Program, 'NASA STI Topics', ftp://ftp.sti.nasa.gov/pub/scan/SCAN-TOPICS

25. National Advisory Committee for Aeronautics (NACA) Report Server. http://www.larc.nasa.gov/naca/

26. 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 and A. Accomazzi, 'The NASA Technical Report Server', *Internet Research: Electronic Networking Applications and Policy*, 5(2), 25-36, (1995).

27. M. L. Nelson , G. L. Gottlich and D. J. Bianco, 'World Wide Web Implementation of the Langley Technical Report Server', NASA TM-109162, September 1994.

28. M. L. Nelson, K. Maly and S. N. T. Shen, 'Building a Multi-Discipline Digital Library Through Extending the Dienst Protocol', *Proceedings of the Second ACM International Conference on Digital Libraries*, pp. 262-263, Philadelphia, PA, July 20-23, 1997.

29. M. L. Nelson, K. Maly and S. N. T. Shen, 'Buckets, Clusters, and Dienst', Old Dominion University, Computer Science Technical Report 97-30, May 1997.

30. M. L. Nelson and S. L. Esler, 'TRSkit: A Simple Digital Library Toolkit', *Journal of Internet Cataloging*, 1(2), 41-55, 1997.

31. A. M. Odlyzko, 'Tragic Loss or Good Riddance? The Impending Demise of Traditional Scholarly Journals', *International Journal of Human-Computer Studies*, 42, 71-122, 1995.

32. D. A. Patterson, 'How to Have a Bad Career in Research/Academia', Keynote Address at the First Symposium on Operating System Design and Implementation, Monterey, CA, November 14-17, 1994. http://http.cs.berkeley.edu/~patterson/talks/bad.ps

33. D. G. Roper, M. K. McCaskill, S. D. Holland, J. L. Walsh, M. L. Nelson, S. L. Adkins, M. Y. Ambur and B. A. Campbell, 'A Strategy for Electronic Dissemination of NASA Langley Technical Publications', NASA TM-109172, December 1994.

34. B. Schatz and H. Chen, 'Building Large Scale Digital Libraries', *IEEE Computer*, 29(5), 22-26, (1996).

35. B. Schatz, W. H. Mischo, T. W. Cole, J. B. Hardin, A. P. Bishop, and H. Chen, 'Federating Diverse Collections of Scientific Literature', *IEEE Computer*, 29(5), 28-36, (1996).

36. E. W. Scott, 'New Patterns in Scientific Research and Publication', *American Documentation*, 4(3), 90-95, (1953).

37. J. Sobieszczanski-Sobieski and R. T. Haftka, 'Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments', 34th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, AIAA Paper No. 96-0711, January 15-18, 1996.

38. I. Stern, 'Scientific Data Format Information FAQ', 1995. http://www.cv.nrao.edu/fits/traffic/scidataformats/faq.html

39. J. Sobieszczanski-Sobieski, 'A Proposal: How to Improve NASA-Developed Computer Programs', NASA CP-10159, 1994, pp. 58-61.

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

41. M. Van Alstyne and E. Brynjolfsson, 'Could the Internet Balkanize Science?', *Science*, 274, 1479-1480, (1996).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
August 1997

3. REPORT TYPE AND DATES COVERED
Technical Memorandum

4. TITLE AND SUBTITLE
Building Multi-Discipline, Multi-Format Digital Libraries Using Clusters and Buckets

5. FUNDING NUMBERS

6. AUTHOR(S)
Michael L. Nelson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-112876

11. SUPPLEMENTARY NOTES
Also published as a MS Thesis for the Old Dominion University Computer Science Department.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited
Subject Category 82 Distribution: Nonstandard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
Our objective was to study the feasibility of extending the Dienst protocol to enable a multi-discipline, multi-format digital library. We implemented two new technologies: cluster functionality and publishing buckets. We have designed a possible implementation of clusters and buckets, and have prototyped some aspects of the resultant digital library. Currently, digital libraries are segregated by the disciplines they serve (computer science, aeronautics, etc.), and by the format of their holdings (reports, software, datasets, etc.). NCSTRL+ is a multi-discipline, multi-format digital library (DL) prototype created to explore the feasibility of the design and implementation issues involved with created a unified, canonical scientific and technical information (STI) DL. NCSTRL+ is based on the Networked Computer Science Technical Report Library (NCSTRL), a World Wide Web (WWW) accessible DL that provides access to over 80 university departments and laboratories. We have extended the Dienst protocol (version 4.1.8), the protocol underlying NCSTRL, to provide the ability to cluster independent collections into a logically centralized DL based upon subject category classification, type of organization, and genre of material. The concept of buckets provides a mechanism for publishing and managing logically linked entities with multiple data formats.

14. SUBJECT TERMS
WWW, Digital Libraries, STI, Distributed Information Retrieval

15. NUMBER OF PAGES
62

16. PRICE CODE
A04

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	