TRSkit: A Simple Digital Library Toolkit

Michael L. Nelson
<m.l.nelson@larc.nasa.gov>

Sandra L. Esler
<s.l.esler@larc.nasa.gov>

NASA Langley Research Center
MS 157A
Hampton, VA 23681-0001

Abstract

This paper introduces TRSkit, a simple and effective toolkit for building digital libraries on the World Wide Web. The toolkit was developed for the creation of the Langley Technical Report Server and the NASA Technical Report Server, but is applicable to most simple distribution paradigms. TRSkit contains a handful of freely available software components designed to be run under the UNIX operating system and served via the World Wide Web. The intended customer is the person that must continuously and synchronously distribute anywhere from 100 - 100,000's of information units and does not have extensive resources to devote to the problem.

Journal of Internet Cataloging, 1(2), 1997, pp. 41-55.

TRSkit: A Simple Digital Library Toolkit

Michael L. Nelson <m.l.nelson@larc.nasa.gov>
Sandra L. Esler <s.l.esler@larc.nasa.gov>

NASA Langley Research Center
MS 157A
Hampton, VA 23681-0001

January 31, 1997

Abstract

This paper introduces TRSkit, a simple and effective toolkit for building digital libraries on the World Wide Web. The toolkit was developed for the creation of the Langley Technical Report Server and the NASA Technical Report Server, but is applicable to most simple distribution paradigms. TRSkit contains a handful of freely available software components designed to be run under the UNIX operating system and served via the World Wide Web. The intended customer is the person that must continuously and synchronously distribute anywhere from 100 - 100,000's of information units and does not have extensive resources to devote to the problem.

1.0 Introduction

This paper introduces TRSkit, a simple toolkit and methodology for quickly implementing a digital library using the World Wide Web as the transmission medium. This design is based on successful architecture implemented in the Langley Technical Report Server (LTRS) (Nelson & Gottlich, 1994; Nelson, Gottlich, & Bianco, 1994) and the NASA Technical Report Server (NTRS) (Nelson, et. al., 1995), but may be shaped to fit most distribution paradigms. TRSkit is designed to be simpler to install and operate than similar systems, such as the Dienst protocol (Davis, Krafft, & Lagoze, 1995).

The intended customer is not necessarily the information science researcher, but rather the individual that is responsible for the dissemination of a body of information units and does not have a lot of time or money to devote to the task. Therefore, this scheme is specifically aimed at government research laboratories, universities and other organizations that desire to distribute their information and do not have distribution limitations or chargeback considerations. This model could be grown to incorporate payment and security, but it is not addressed at this time. The software introduced in this paper is available at:

```
http:/techreports.larc.nasa.gov/ntrs/xtrs.html
```

2.0 Guidelines

Some basic presuppositions guided development. Understanding these provides insight to design decisions made during the development of TRSkit.

2.1 Simple things should be done simply

The requirement addressed is to set up a digital library that distributes documents with minimum difficulty. Keep in mind the customer is there to access and retrieve an information unit (report, data set, image, software package, etc.) in a convenient and timely manner.

2.2 Focus on distribution only

Organizations often tackle too large a problem and attempt to address editing, routing, approval, scanning, etc. in their digital library projects. While these areas are interrelated, there is considerable utility in initially isolating them and resisting broad brush, "one-size fits all" solutions. What is good internally for information production or collection management may not be how the intended customers wish to access data.

2.3 Deliver the final product, not just meta-data

Unless distribution is to be controlled for reasons of security or chargeback, avoid wasting time distributing abstracts or other meta-data. Concentrate only on delivering the final product (full publication, software package, etc.) to the customer.

2.4 Use the World Wide Web for distribution

Using the web seems obvious now, but used to be a more controversial point. In the past, many database and digital library projects failed because they spent most of their resources defining the specialized software for access, and none of their resources for actually populating their database.

Specialized client development is justified only when the application requirements are so specialized that they cannot be satisfied through a general web browser. The increasing use of Java (Arnold & Gosling, 1996) and Tcl/Tk (Ousterhout, 1993) for extending browser functionality should make specialized client development even more rare.

2.5 Set up your digital library in a proper environment

This includes both the development and production environment. Historically, this has meant using Unix, however Windows NT has presently duplicated most of the useful Unix functionality and could possibly be used as well (note that the tools presented within this paper have only be tested on a Unix platform). Be sure that the machine chosen is correctly configured to withstand the increased load of an active digital library.

3.0 Background

TRSkit is stable and mature enough to form the foundation of popular production systems. The components described here were assembled beginning in 1992 starting with the creation of the Langley Technical Report Server. The follow-on project of the NASA Technical Report Server produced even more components. LTRS has been in operation since January 1993, and NTRS has been in operation since June 1994. Usage has continually increased over time, with NTRS averaging over 30,000 searches a month from sources world wide (Figure 1).

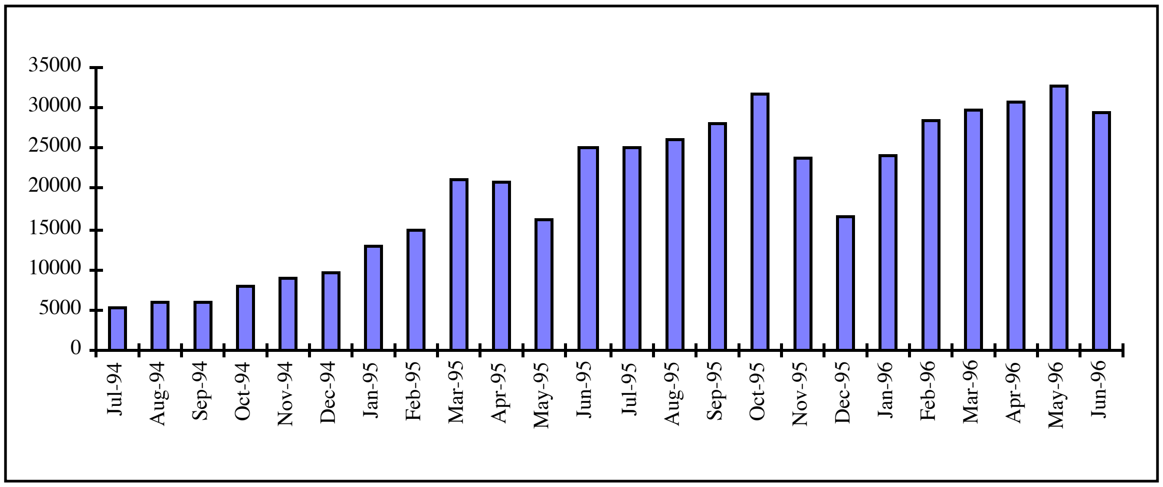

Figure 1. Average Usage of the NASA Technical Report Server.
(Note: NTRS was not available during the US Government Furloughs of November and December 1995)

It is important to note that the toolkit is not only useful to publications servers (Table 1). It has been demonstrated for photo, fact sheet, and software servers.

•	Photograph archives
•	Video, movie archives
•	Software servers
•	Fact sheets archives
•	Music archives
•	Electronic Forms
•	News Articles
•	Product Catalog
•	Expert Systems

Table 1. Examples of distribution servers that TRSkit can support.

4.0 Architectures

The architecture is composed of an underlying framework wrapped with interfaces to hide the details of the system from the user. Hence, there are two distinct points of view: the user's and the maintainer's.

4.1 User's Point of View

Figure 2 illustrates how the user perceives the system. Options for browsing and keyword searching are implemented. Browsing provides a method of perusing meta-data, such as abstracts and citations, with links to the appropriate final products (publications, software packages, etc.). Keyword searching provides similar access. A user enters the keywords, a search is performed against the indexed meta-data, and a list of titles returned. The title then lead to a meta data page, with the final product linked from there.

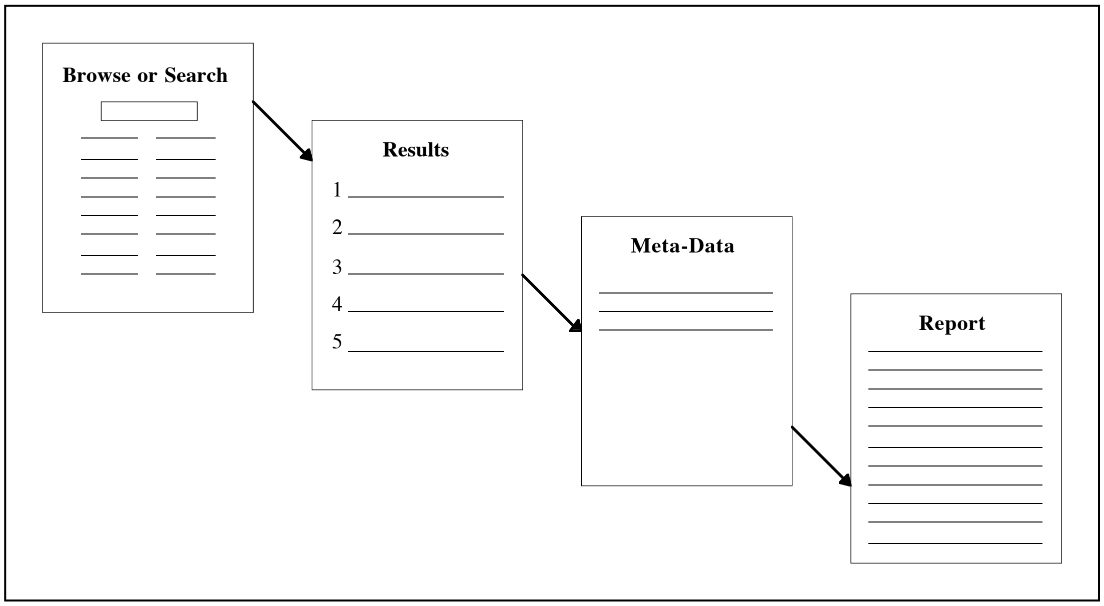

Figure 2. How the users perceives the system.

The intent is to offer the user a lightweight mechanism to refine their search / browsing by offering increasing levels of description about the final product, which is assumed to be a heavier weight in downloading, either in time to retrieve, or even chargeback.

4.2 Maintainer's Point of View

The maintainer is responsible for tracking two items: the meta-data and the actual data item. A simplifying assumption is made that the actual data item cannot be searched or otherwise have its information extracted. For instance, the data item may be an image, executable software, or an arbitrary document type. Experience from LTRS has indicated that users do not mind this restriction.

The logical organization that the maintainer keeps does not have to be the same one that the user sees. Thus, the data items are free to be files in a file system, or objects in a database. The meta-data could be either as well, but in practice, it is easier to keep meta-data as a collection of files.

Note that the meta-data is not archived in HTML. There is great utility in archiving the meta-data in a structured format that is presentation independent. It is trivial to convert the meta-data to HTML either on the fly or in advance.

5.0 Components

TRSkit is composed of the following set of freely available components designed to create an efficient digital library system (Figure 3):

- Meta-Data Processing Tool
- Report Object Management
- Search Engine
- Administration Scripts
- UNIX and Perl

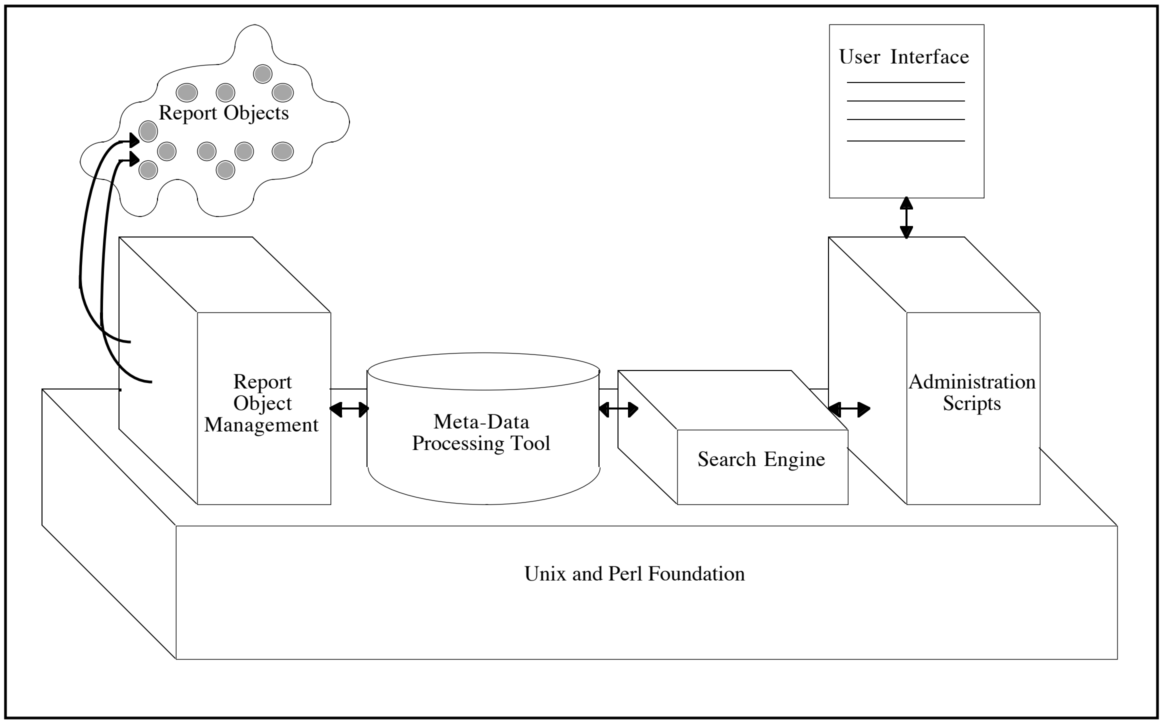

Figure 3. Intra-componet communications.

5.1 Meta-data Processing Tool

In digital libraries, the actual data items do not get manipulated by the maintainer. Often they cannot be indexed or searched because they are images, executable software, or other difficult to abstract data formats. Thus only a tool for manipulating meta-data is employed. The tool used is a modified public domain Perl script, which converts citations and abstracts in bib/refer (Lesk, 1978), bibtex (Lamport, 1986), RFC-1807 (Lasher & Cohen, 1995), and other formats into HTML. Figure 4 contains an example of meta-data used in TRSkit, which is predominately in refer format.

```
%T A Parallel Prefix Algorithm for Almost Toeplitz Tridiagonal Systems
%A Xian-He Sun
%A Ronald D. Joslin
%D December 1995
%P 547-576
%K High performance computing; Parallel numerical algorithm; Tridiagonal system
%I NASA Langley Research Center, Hampton, VA, 23681-0001
%J International Journal of High Speed Computing
%V 7
%N 4
%O (185KB)
%U ftp://techreports.larc.nasa.gov/pub/techreports/larc/95/NASA-ijhsc-95-p547.ps.Z
%X A compact scheme is a discretization scheme that is advantageous in obtaining highly accurate solutions.  However, the
resulting  systems from compact schemes are tridiagonal systems that are difficult to solve efficiently on parallel computers.
Considering the almost symmetric Toeplitz structure, a parallel algorithm, simple parallel prefix (SPP), is proposed.  The SPP
algorithm requires less memory than the conventional LU decomposition and is efficient on parallel machines.  It consists of a
prefix communication pattern and AXPY operations.  Both the computation and the communication can be truncated without
degrading the accuracy when the system is diagonally dominant.  A formal accuracy study has been conducted to provide
a simple truncation formula.  Experimental results have been measured on a MasPar MP-1 SIMD machine and on a Cray 2 vector
machine.  Experimental results show that the simple parallel prefix algorithm is a good algorithm for symmetric, almost symmetric
Toeplitz tridiagonal systems and for the compact scheme on high-performance computers.
```

Figure 4. Example of a meta-data item in refer format.

The key responsibility of the meta-data is to store and link the appropriate data items. Related to this is the decision of choosing a structured format for the meta-data. There is no "right" format, but it is usually advantageous to choose a common format because tools for conversion are likely to be written.

It is important to note that refer supports multiple data formats (i.e., PostScript, PDF, HTML, GIF, etc.). The “%U” tag is used by default to denote the URL of the file, but other fields are defined to handle specific data type:

```
%U-text/html
%U-application/pdf
%U-application/postscript
```

Using MIME types (Borenstein & Freed, 1993) allows a clear path for future data object formats. If these fields are defined in the refer citation then an extra line with “Formats(s):” will appear with the options that are available.

5.2 Report Object Management

TRSkit can support several models of meta-data and report (or other data object) relations. Originally, LTRS began with 1 meta-data object referenced 1 report object, generally a compressed PostScript file. With the increased popularity of Adobe Acrobat (PDF), many meta-data objects now refer to multiple format types. This lead to the MIME type extensions introduced above.

Currently, LTRS and other services maintain both their meta-data and their report objects as files in a standard Unix filesystem. Migrating the report objects to reside in a true database would not greatly impact TRSkit since the %U field can be used to “point” to objects within a database. Storing the meta-data itself in a database is possible, but would require more modifications to the meta-data processing tools.

5.3 Search Engine

A search engine extracts or filters information from the data collection, transforms it into an index format, interprets the user’s request, and produces a summary of matches relevant to the search term discovered within the index. As determined by the NASA Indexing Benchmarks, search engines can be grouped into two levels according to their performance characteristics (Esler & Nelson, 1996). Level one engines are efficient on small to medium sized data collections. Level two search engines are ideal for data collections up to and beyond 100MB. Table 2 contains recommended freely available (non-commercial) search engines.

Engine	Level	URL
freeWAIS-sf	2	http://ls6-www.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf/freeWAIS-sf.html
FFW	1	http://eminfo.emc.maricopa.edu/ffw/ffwhome.html
Harvest/GLIMPSE	2	http://harvest.cs.colorado.edu
Isite	1	http://www.cnidr.org/Software/software.html
SWISH	1	http://www.eit.com/software/swish/swish.html

Table 2. Recommended freely available search engines.

Each of the above search engines are simple to implement and require minimal customization. Although the level one search engines are superb for small libraries and personal file systems, they are not suggested for use with large scale distribution. Unless the domain of the information within your digital library is restricted, plan for success and choose a level two search engine to insure a framework that is expandable. Thus, the implementation of freeWAIS-sf or Harvest/GLIMPSE is suggested for use with most digital libraries.

5.4 Administration Scripts

Administration scripts provide a backbone for efficient operation within digital libraries. Scripts included in the TRSkit are capable of the following automated tasks:

- Database and usage graphs generated from logfile interpretations.
- Updating HTML interfaces according to objects in the data collection
- Forms to add, edit, and remove objects by members of the administrative team.
- Forms to set search defaults and other options to provide guidance for the novice user.

These scripts play an essential role in the maintenance performance of a digital library. As the data collection grows, it becomes necessary to implement more automated scripts to ensure the timely up keep of the system. Scripts included in this toolkit can be run interactively or out of cron.

5.5 Unix and Perl

Although it is possible to set up a digital library on a PC or Macintosh, TRSkit services to date have been set up on Unix workstations. TRSkit has not been tested on non-Unix machines.

Perl is an interpreted language similar to C and C++ that was created to parse arbitrary files, extract information from them, and manipulate the information into reports. Thus, it works efficiently with digital libraries and system management in general. Perl can be obtained free of charge at:

```
ftp://ftp.netlabs.com/pub/outgoing/perl5.0
```

6. Implementation

Implementation of TRSkit is a simple process. To start with it is necessary to have a WAIS URL that points to the abstract database and a URL that points to the query interface. Second, a version of WAIS must be installed (preferably freeWAIS-sf). Other search engine can be used in place of this, but may require more customization. Third, the citations and abstracts must be put in the chosen meta-data structured format. Using refer is favored, but other formats can be used after some customization of the administrative scripts. Detailed information discussing implementation and source code is available at:

```
http://techreports.larc.nasa.gov/ntrs/xtrs.html
```

7. Future improvements

After establishing the core functionality, new services can be added to the digital library. These could include adding caching optimizations, different front ends or options for beginning and advanced users, or storage side enhancements. One improvement underdevelopment is adding the hooks necessary to support fielded searches where available.

Most importantly, these improvements can be added incrementally as resources allow or requirements demand, while the system continues to serve. The modular nature of these components allow additions and upgrades without disturbing the larger system.

8. Lessons learned

When designing a digital library, there are a number of issues to consider. Most importantly, although this may start out as an expected low volume server, prepare for success. Choose an architecture that will easily expand to accommodate a popular resource. Additional lessons learned with the design and implementation of the TRSkit include:

- Scaleable naming scheme
- Never throw away source to previous formats
- Use rapid prototyping

8.1 Scaleable Naming Scheme

Choose a scaleable naming scheme on the implementation side. For instance, the turn of the century poses a significant challenge to the computing industry because many software designers neglected to prepare for the year 2000 when they chose to use a two digit labeling scheme for years (i.e., 96 verses 1996). Secondly, it is important to include the institution in the naming of the data. For example, using TR-1997-123.pdf may be sufficient from a maintainers point of view, but using MIT-EE-TR-1997-123.pdf is better.

Additionally, if the report objects are going to be stored on a filesystem, choose a properly structured hierarchy that can accommodate thousands of report objects. A file path similar to:

```
.../report-server/institution/department/year/file.format
```

Is preferable to the more often used:

```
.../report-server/reports/file.format
```

8.2 Retain Original Source

Never throw away the original format or source for the data items. Present formats may fail and be less productive then the previous versions, and storage is much cheaper than regenerating the data item. The original format (Word, WordPerfect, TeX, etc.) does not have to be served to the public, but it keeping it increases flexibility for future conversion efforts.

8.3 Rapid Prototyping

Use rapid prototyping: start small and get customer feedback early. Many digital library and data management projects never get off the ground because the sheer size of their legacy information problems. Do not let that dissuade you from providing meaningful, albeit small, services today (Figure 5).

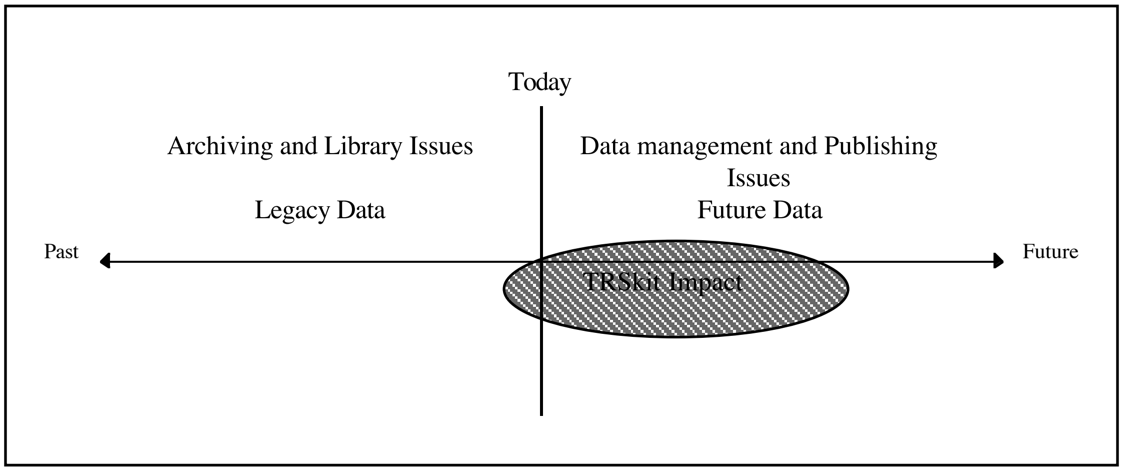
Today
Archiving and Library Issues
Legacy Data
Data management and Publishing Issues
Future Data
Past
Future
TRSkit Impact

Figure 5. Past and Future Issues in publishing and distribution.

The size of the archiving and library issues should not interfere with the data management and publishing issues; they are parallel activities. Coordination between the activities is required, however, in a limited resource environment, an adequate digital library can be built using TRSkit to sample the existing publishing process and populate the collection.

9. Conclusion

Large volume, customer pleasing digital libraries are not the strictly the realm of complex research prototypes or expensive commercial systems. The presence of the universal information client, the WWW browser, greatly simplifies providing access to multimedia digital libraries. Organizations that have limited resources yet need to distribute their information collection in a scaleable manner can use TRSkit to build a extensible digital library.

TRSkit has been in use in the many servers that make up the NASA Technical Report Server and other resources since 1994. The collection of tools that make up TRSkit consist of freely available components that provide for easy distribution on the World Wide Web. Further information and source code is available at:

`http://techreports.larc.nasa.gov/ntrs/xtrs.html`

References

Arnold, Ken, & Gosling, James. (1996). The Java Programming Language. Reading, MA: Addison-Wesley.

Borenstein, N. & Freed, N. (1993). MIME (Multipurpose Internet Mail Extensions) Part One: Mechanism for Specifying and Describing the Format of Internet Message Bodies. Internet RFC-1521.

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

Esler, Sandra L. & Nelson, Michael L. (1997). A Comparison of Performance Characteristics of Freely Available Index and Search Engines. To Appear in *The Journal of Network and Computer Applications*

Lamport, Leslie (1986). LaTeX: A Document Preparation System. Addison-Wesley.

Lasher, R. & Cohen, D. (1995). A Format for Bibliographic Records. Internet RFC-1807.

Lesk, M. E. (1978). Some Applications of Inverted Indexes on the UNIX System. Computing Science Technical Report 69. Murray Hill, NJ: Bell Laboratories.

Nelson, Michael L. & Gottlich, Gretchen L. (1994). Electronic Document Distribution: Design of the Anonymous FTP Langley Technical Report Server. NASA TM-4567. Hampton, VA: NASA Langley Research Center.

Nelson, Michael L., Gottlich, Gretchen L., & Bianco, David, J. (1994) World Wide Web Implementation of the Langley Technical Report Server. NASA TM-109162. Hampton, VA: NASA Langley Research Center.

Nelson, M. L., Gottlich, G. L., Bianco, D. J., Paulson, S. S., Binkley, R. L., Kellogg, Y. D., Beaumont, C. J., Schmunk, R. B., Kurtz, M. J., Accomazzi, A. & Syed, O. (1995). The NASA Technical Report Server. *Internet Research: Electronic Network Applications and Policy*, 5(2), pp. 25-36.

Ousterhout, J. K. (1993). An Introduction to Tcl and Tk. Reading, MA: Addison-Wesley.