The Networked Computer Science Technical Report Library

James R. Davis[*] Carl Lagoze[†]

Submitted to May 1996 IEEE Computer special issue on "Building Large-scale Digital Libraries".

Abstract

The Networked Computer Science Technical Report Library (NCSTRL[1]) is a distributed digital library of research results from computer science departments and laboratories in the USA and abroad. NCSTRL benefits readers, authors, and departments. Researchers throughout the world can use familiar Internet tools (the World Wide Web) to search for, browse, read, and download technical reports from participating institutions. Authors - benefit by reaching a wider audience. Departments gain a clean, effective management system for distributing their technical reports and eliminate much of their current copying and mailing charges. This article describes the design of NCSTRL, its historical basis in earlier work, and the expected course of its development.

Keywords: protocols, document management, digital libraries

Introduction

A good library is a carefully selected collection of materials systematically organized in a manner accessible to users. Computer scientists have long desired to use the Internet as the basis of a library of computer science research. Until recently, sharing of research was usually done either by email or in anonymous FTP archives. Archives, while essential, lack many services required of a library. Over the past few years, three systems have appeared that bring us closer to having a true library of computer science research over the Internet.

The UCSTRI system[7] made FTP archives more useful by collecting indexes of their contents at a centrally searchable site. UCSTRI's weakness was that the descriptions of contents on which it based its search were relatively crude and uncontrolled, usually being "README" files, which are intended for human eyes rather than automated search engines. Besides being unstructured, these files were rarely maintained with care. As a result, UCSTRI often returned imprecise results.

The next two systems, WATERS[5] and Dienst[4] improve on UCSTRI. WATERS mandates a uniform file format (an extended version of refer) for cataloging. This removes a lot of confusing

[*]Xerox Corporation, Design Research Institute, 502 Rhodes Hall, Cornell University, Ithaca NY 14853. davis@dri.cornell.edu

[†]Dept. of Computer Science, Cornell University, Ithaca NY 14853. lagoze@cs.cornell.edu

[1]pronounced "ancestral" and hopefully connoting bountiful progeny

results from searches. WATERS also includes a tool (`techrep`) to help site administrators keep the cataloging files up to date. But this comes at a cost. Sites distributing works with UCSTRI don't have to do anything they aren't already doing - UCSTRI copies the index files itself. Sites running WATERS have to actually install and run some code to participate. WATERS takes more effort to install and run than UCSTRI, but provides better results.

We describe in this paper the Networked Computer Science Technical Report Library, which builds upon Dienst and brings us closer to the goal of a universal computer science research library. We begin by describing Dienst. Then we describe the institutional context of NCSTRL, followed by a description of the new features in NCSTRL. Finally, we describe future work in this area.

Dienst overview

Dienst was developed as part of the ARPA-funded CS-TR project[1], which consisted of five universities (Berkeley, Cornell, CMU, MIT, and Stanford) coordinated by the Corporation for National Research Initiatives (CNRI). The project had two broad goals: develop methods to make computer science research available over the Internet and undertake basic research in digital libraries. The Dienst architecture was developed in response to the first goal. A reference implementation (in perl) was distributed to the five sites. As it became stable, other universities began to run the code.

Dienst improves on previous systems in three ways. First, it specifies a **document services architecture** where logically distinct services are performed by separate modules. Second, it mandates an open, extensible **protocol** for communicating with these services. Finally, it has a structured **document model** that provides for multiple formats and multiple decompositions of the document.

In the document architecture, each document has a unique, location-independent identifier, called a **docid**. A docid has two components: a **publisher** and a **name**. A publisher name corresponds to a participating site, and is assigned by a central Dienst administrator. Each publisher assigns names in any manner it chooses. Since docids are independent of location and format, they are suitable for permanent, archival reference to documents, an essential property for a library not found in earlier systems.

The Dienst architecture specifies three types of digital library services: **repository** services store the document in its many formats, and support retrieval of all or part, **index** services provide search, accepting a query and returning a list of docids matching the query, and **user interface** services use information obtained from the other services and present it in a human readable manner. A fourth kind of service, the **Meta** service, provides contact information for the other three.

Of these four services, only the first is used directly by a human, via a World Wide Web client. The others are used by programs, in particular Dienst services, but also by other digital library or publishing systems. For example, the Stanford Information Filtering Tool[9] (SIFT) obtains the bibliographic records it processes from the index service.

Figure 1 shows how these services cooperate. A user connects to the user interface service, and issues a search request, specifying search keys such as title or abstract words, optionally joined by boolean operators. The UI service sends to the index server of each publisher (in parallel) a search request. Each index server returns a list of docids for documents matching the query, then the UI server formats and presents them to the user as a hypertext list of links. Each link (for a document) leads to the UI server operated by the publisher of that document.

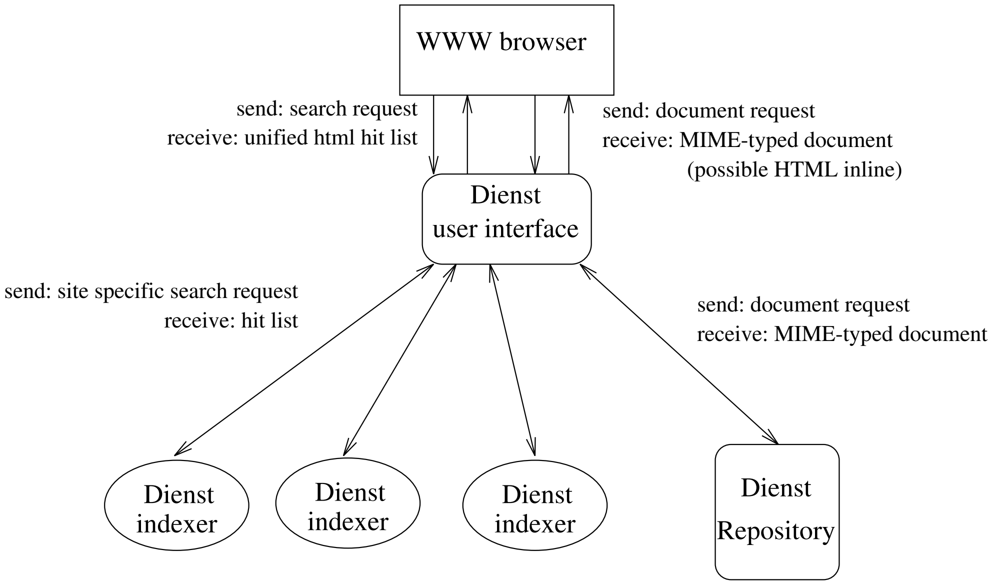

Figure 1: Users of Dienst interact with a User Interface server through a standard Web client. The UI server searches a set of Index servers, and displays the results as a set of hyperlinks. Each link leads to the UI server for that document, which may be on a completely different server.

Although previous systems included components to perform all three of these services, they did not make a clear separation between them. This separation makes possible the specification of a simple, open protocol for the various services, and this in turn enables third party systems (such a SIFT) to use the library. It also allows researchers interested in a single aspect of digital library research to use Dienst as a testbed, substituting a new component (e.g. an indexer) for the one provided in Dienst, while retaining the rest of the system.

Dienst supports a message-passing interface to the architecture. Messages may be addressed to a particular server, to one document, or to a particular part of a document. A message may be sent to any convenient Dienst server (the nearest, for example), which will execute it locally or forward it as appropriate. Dienst thus appears to be a single virtual document collection, and hides the details of the server distribution.

In the current version of Dienst, messages between servers are encoded as URLs and transported with the HTTP protocol, and delivered to Dienst services through the Common Gateway Interface (CGI) of the Web. In a future version, we will use a different transport mechanism.

The Dienst document model (as shown in figure 2) allows documents to be in many formats. Messages to the document allow one to get a list of the available formats, or to obtain a version in a specified format.

A document enters Dienst either by being scanned (as a set of TIFF files) or in electronic format

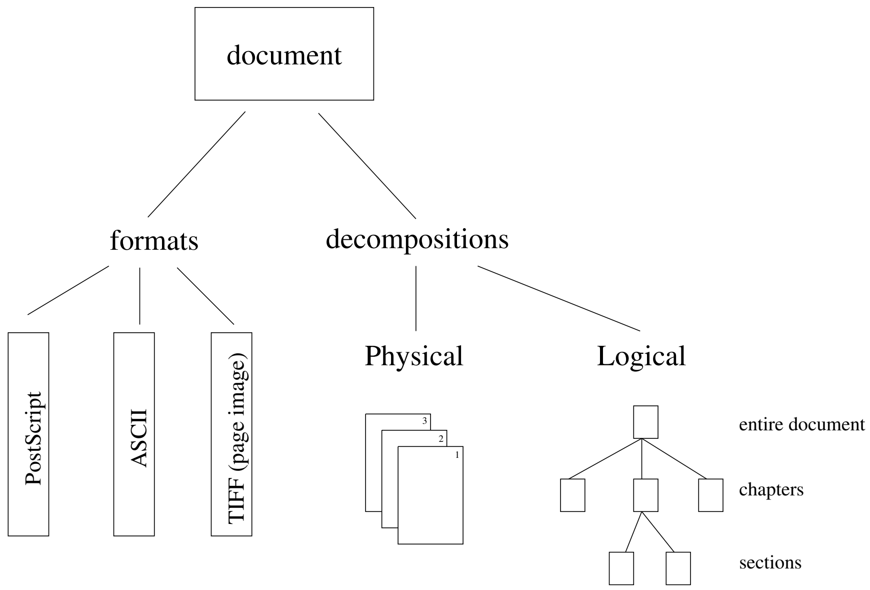

Figure 2: Dienst document model: a document may be stored in multiple formats, and may have multiple decompositions, either physical (e.g. page number) or logical.

(as PostScript). Typically, a site will then generate additional formats for convenience. For example, a scanned format might be converted to plain text by OCR for purposes of full text retrieval, or a set of miniature page images might be generated to support rapid visual browsing. The ability to refer to a document regardless of content, combined with the ability to obtain specific formats as needed, is one of the strengths of Dienst.

NCSTRL Context

The three systems described earlier, UCSTRI, WATERS, and Dienst, successfully evolved the technology for creating an on-line technical report library. Yet all three were unsuccessful in their efforts to grow their collection beyond a small subset of computer science departments. We see a number of reasons for the failure to create a universal computer science technical report library based on any of these technologies.

Each of the systems were initially developed as research projects. From the perspective of most departments this meant that the systems were potentially unstable, subject to frequent and incompatible changes, and lacked any support for users. Furthermore, without organizational backing outside the research group, there was the valid concern that the research projects would end and the developers would move on to other projects.

A related reason for the failure of these efforts is the fact that digital library research has, in general, lacked coordination and standardization. Departments could not see how any of these systems fit into the larger scheme and preferred to wait for the digital library field to define itself before making a commitment to a system. Instead, they often adopted the lowest investment alternative, which meant putting technical reports on an anonymous FTP archive.

A third reason was the failure to adequately publicize the systems. In many cases a department adopted one of the systems after a graduate student discovered it on the Internet. Department chairs or senior faculty were rarely involved in the technology adoption process, further eroding the integrity of the technical report library.

We designed both the technical and organization infrastructure of NCSTRL to overcome the difficulties of the previous efforts. NCSTRL has firm organization backing, has a development plan that is coordinated with national digital library research, and will be publicized to departments at the highest level.

From its inception, NCSTRL has been working with the Computing Research Association (CRA), an association of more than 140 North American computer science departments, industrial laboratories and affiliated professional societies. In July, 1995 CRA endorsed the concept of a national digital library of computer science technical reports and has recommended participation in the project by its member institutions. CRA has strong links to department chairs and this endorsement lends credibility and stability to the project. We will coordinate with CRA to publicize NCSTRL at conferences and in publications. NCSTRL has also been endorsed by the SURA/SOLINET Working Group on Electronic Theses, Dissertations and Technical Reports. The membership of SURA/SOLINET includes most academic institutions in the southeast United States. The endorsement by SURA is a model for participation by other regional organizations.

NCSTRL is coordinated with national digital library research at two levels. Organizationally, NCSTRL development and policy is overseen by the NCSTRL working group (NCSTRL-WG). This group exists within the ARPA-funded Digital Library Forum (D-Lib Forum). Led by the Corporation for National Research Initiatives (CNRI), D-Lib Forum undertakes activities to facilitate the transfer of research into the creation of a national digital library system. The existence of NCSTRL-WG within this structure will ensure that evolution of NCSTRL takes place in this larger context. Technically, we are committed to using portions of the digital library infrastructure as they develop. We are already using handles and our plans to evolve NCSTRL to a more open repository architecture are described below.

NCSTRL technical overview

While the NCSTRL architecture is based upon Dienst, the NCSTRL system as a whole improves upon Dienst in several ways:

- It uses handles for location-independent document naming.
- It tolerates failure of index servers by providing a backup server.
- It allows sites to choose from two levels of software, NCSTRL *Standard* and NCSTRL *Lite*. The latter has fewer features, but is easier to install and less demanding of system resources.
- It provides tools to automate library management.

We now examine each of these.

Handles provide location-independent naming

Dienst's method for location independent naming, docids, has two problems. First, the name space it provides does not scale well. A docid consists of a publisher and a publisher-assigned name. This provides a single, flat namespace of publishers, which, while probably sufficient for NCSTRL (which will probably never have more than 200 publishers), would be unbearable in a universal library. Since we hope that someday NCSTRL will be part of a larger collection, we need a more flexible naming system. The Dienst resolution mechanism is internal to the system and is not available through the protocol. This makes portions of the Dienst architecture inaccessible to outside systems.

The NCSTRL system will use the Handle System for document naming. The Handle System was developed by the Corporation for National Research Initiatives (CNRI) as part of the Computer Science Technical Reports project. Fundamentally, a handle is an identifier with two components, a naming authority and a string. The naming authority is similar to the publisher in a docid, but is a hierarchical name space rather than a flat namespace of publishers, and thus provides for expansion. The string of a handle, like the name in a docid, is assigned by the naming authority, and guaranteed unique. The translation between Dienst docids and NCSTRL handles is straightforward. A docid issued by publisher P with name n becomes the handle `ncstrl.`P/n.

Each handle has one or more fields of typed data, which may store arbitrary data. To resolve a handle is to present a handle to the system and have returned some or all of the fields For purposes of NCSTRL, we will use one of the fields to store the location of User Interface server for the document. In the future we may extend our use of the Handle System to refer to each format available for a document.

The handle system includes a handle server that resolves handles. The server is a distributed system that is publicly accessible, highly secure, fault tolerant, and designed to run continuously. Using the handle server as the infrastructural base for NCSTRL provides a base for future expansion and makes possible future interoperation with other digital library projects that also use the handle system.

Backup servers provide fault tolerance.

In our experience with Dienst, it was rare for a search to successfully contact all the index servers. There were many causes for the failures, including network failure (e.g. fiber cable cuts), sites bringing down their servers, and servers being too heavily loaded. Such failures were not catastrophic, because they timed out, but they did slow system response, and made searching unpredictable. The same search on different days could yield very different responses. Some would claim that this is an argument against distributed indexing, but we believe that in the long run, distributed indexing is essential, because it scales well, allows for site specific indexing, and protects intellectual property (since indexing information itself may be copyrighted).

The NCSTRL system tolerates index server failure by using backup index servers. For each publisher it is possible to declare a (sorted) list of index servers, instead of just one. The first server is the primary server, and the others are secondary, tertiary, etc. Backup servers contain copies of index records from the primary server. Should the primary server fail to respond, NCSTRL contacts the backup. If the primary continues to fail for a certain number of times, it is marked as down, and not visited again for a fixed time. This approach saves the time that would have been expended waiting for the server to timeout. This same idea allows us to service sites that are relatively far from the NSF backbone (such as in Europe) by placing a backup server nearer to them.

NCSTRL *Lite* allows sites with fewer resources to participate.

For some sites, the decision whether to participate in a library project depends strongly on the difficulty of installing the software. While we hope that the perceived benefit of NCSTRL will be larger than that of Dienst or WATERS, we also recognize that not all sites will be able to to install a system like Dienst. Therefore we've adopted an approach that allows sites to participate in NCSTRL, albeit at a lower level of functionality, by installing what we call NCSTRL *Lite*. The *Lite* package is quite similar to WATERS, in that sites running NCSTRL *Lite* store their documents in FTP archives, not Dienst repositories, and create bibliographic records with the `techrep` tool. Where WATERS uses a central WAIS index, which also acts as the user interface server, NCSTRL *Lite* uses a distinguished central index server running the Dienst protocol. This server periodically polls Lite sites to copy bibliographic records, and in effect provides a full Dienst protocol gateway into the Lite sites.

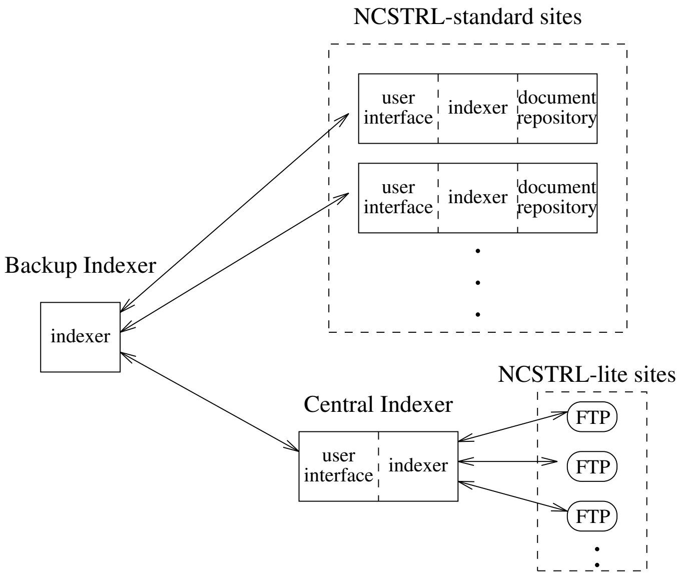

Figure 3: NCSTRL topology: NCSTRL *Standard* servers support three services (UI, Index, and Repository). *Lite* sites run a simple FTP server, and a dedicated Central Indexer provides UI and Index services on their behalf. A Backup server mirrors contents of all Index servers.

The distinction between the central or distributed indexes and between FTP repositories or NCSTRL repositories is invisible to the user. All documents from all institutions are searchable through any of the user interface servers. The results of searches are selectable links to the repository copies of documents that match the search criteria. The interface for documents resident in a NCSTRL server repository or an FTP repository site is exactly the same.

Library management tools simplify administration.

Sites require good tools for maintaining their bibliographic databases. The person responsible at a site is often either a volunteer (with little time to spare) or an administrative assistant (with little technical sophistication). Therefore it's important to make this task as easy as possible. WATERS provided a tool, `techrep` to help edit the bibliography files, and we also include this for sites running NCSTRL *Lite*.

Sites running NCSTRL *Standard* have more data, and therefore require more tools to manage it. Most of this extra data is for the alternative formats of the document, for example the reduced size page images for visual browsing. NCSTRL *Standard* includes a tool `db_build`, that automatically constructs such formats from the original document images. This makes it easier for sites to take advantage of the advanced user interface features of NCSTRL *Standard*.

NCSTRL *Standard* provides additional tools to automate some of the mundane aspects of maintaining a departments document collection:

- While NCSTRL removes much of the need for printing, there are still occasional requirements to generate paper copies, either for archival purposes or for readers who are not on the Internet. NCSTRL provides a tool to generate a paper copy, complete with formatted cover page.

- Many departments generate periodic summaries of newly issued reports. NCSTRL provides a tool to automate construction of such summaries.

- It is sometimes necessary to remove a technical report from the online collection, either because the rights to it have been transferred to another entity, or because of a flaw. NCSTRL provides a tools to automate such withdrawal.

While none of these tools are individually very important, we think that together they make NCSTRL more attractive, and hence help us to reach critical mass.

Future Work

A significant benefit of the NCSTRL project is the foundation it provides for a variety of future research and development projects. These projects fit into three classes: work that is internal to the NCSTRL project and architecture, research and development outside the project that uses the infrastructure provided by NCSTRL, and efforts to evolve NCSTRL in parallel with national digital library efforts.

The NCSTRL system as of November, 1995 lacks a number of desirable features that affect its robustness, performance, and functionality. We plan a number of projects to address these shortcomings.

One area we plan to examine are methods to improve the robustness of distributed search. The current system mirrors index sites at a single backup server. Reliability would be improved by having multiple mirrors and developing methods for dynamically routing search requests to the currently most available server. Similar methods could be used to mirror repositories.

Another problem with the current system is the inability to deliver partial results. A distributed search only returns after all sites have responded, or a timeout occurs. We are currently limited in this area by the technology of the Web, but expect this to change in the near future so that we may develop methods to incrementally deliver results.

Finally, there are two interesting projects that may be pursued through a developing partnership with the European Research Consortium for Informatics and Mathematics (ERCIM), which represents 14 national information technology research institutes. The first project involves extending the system to deal with multiple languages, both at the user interface and search engine level. This is a topic that must be examined for the internationalization of digital libraries, and NCSTRL provides a perfect platform on which to prototype work in this area. A second project of interest in the international context is developing methods for dealing with low-bandwidth networks. What tools can we develop to make an on-line collection accessible to users using low-speed connections? Obviously, the full functionality of the library will not be available to these users, but some usable subset should be.

The NCSTRL collection and infrastructure will be a valuable resource to researchers not directly involved in the project. The open NCSTRL protocol permits the development of other tools that access the elements of the collection for analysis and experimentation. We expect that researchers in areas such as natural language understanding, information retrieval and discovery, and user interface tools for browsing digital libraries will use the NCSTRL collection as a testbed for their work.

Finally, as mentioned earlier, we are committed to evolving NCSTRL in the context of the development of a national digital library infrastructure. The CS-TR project produced two important elements of this infrastructure. The first is the handle system, described earlier and used in NCSTRL. The second is an open architectural framework for distributed repositories containing mixed-content digital objects[2]. This is commonly referred to as the Kahn/Wilensky architecture, after its authors Robert Kahn (CNRI) and Robert Wilensky (UC Berkeley). An important contribution of this architecture is the attention it pays to copyright issues such as preservation of intellectual property rights for digital objects.

Future versions of NCSTRL will be based on this architectural framework as its design evolves and aspects of it are implemented. In parallel with the development of NCSTRL, we have been involved over the past year in articulating the implementation issues of the Kahn/Wilensky architecture[3]. At present, we are collaborating with researchers at NCSA and CNRI on the detailed definition of an architecture for interoperating secure object stores, which is based on Kahn/Wilensky. In this architecture, each object store is a repository for typed digital objects that have globally unique names and which may have arbitrarily complex terms and conditions that govern access, mutation, management, and deletion of the object. An object store is responsible for controlling client interaction with its contained objects as defined by their respective terms and conditions. Because the design of this architecture is based upon the idea of distributed objects, we plan to implement prototypes of this system using a CORBA-based[6] system such as ILU[8].

NCSTRL will provide a foundation to experiment with the robustness and interoperability of this new architecture.

Conclusion

At the time this article was written, NCSTRL was running in alpha test. But by the time it is printed, it should be running and stable. We are eager to add more departments to the collection. Participation is open to all departments awarding a PhD in computer science or engineering and to research facilities of industry and government. We also expect to support significant international participation. For further information, see the NCSTRL home page at `http://www.ncstrl.org/info/about-ncstrl.html`.

Acknowledgments

The authors thank William Arms of CNRI for information about the handle system. We also thank all our collaborators on the NCSTRL project (Ed Fox, Jim French, Dean Krafft, Rebecca Lasher, Kurt Maly, and Alan Selman). None of these people, nor their institutions, are in any way responsible for any errors in this paper.

This work was supported in part by the Advanced Research Projects Agency under Grant No. MDA972-92-J-1029 with the Corporation for National Research Initiatives (CNRI) and by the National Science Foundation under Grant No. NSF-CDA-9308259. Its content does not necessarily reflect the position or the policy of the Government or CNRI, and no official endorsement should be inferred.

References

[1] Corporation for National Research Initiatives, "CSTR Computer Science Technical Reports", `http://WWW.CNRI.Reston.VA.US/home/cstr.html`.

[2] Kahn, R. and R. Wilensky, "A Framework for Distributed Digital Object Services", `URL: http://WWW.CNRI.Reston.VA.US/home/cstr/arch/k-w.html HDL:cnri.dlib/tn95-01`

[3] Lagoze, C. and D. Ely, "Implementation Issues in an Open Architectural Framework for Digital Object Services", Technical Report TR95-1540, Department of Computer Science, Cornell University.

[4] Lagoze, C. and J. R. Davis, "Dienst: An Architecture for Distributed Document Libraries," *Communications of the ACM*, 38(4), April 1995, p. 47.

[5] Maly, K., J. C. French, A. Selman, E. A. Fox, "Wide Area Technical Report Service," *2nd International World Wide Web Conference, WWW'94* Oct. 1994, pp. 523-533.

[6] Object Management Group, "CORBA and the OBM", URL: http:////www.acl.lanl.gov/sunrise/DistComp/Objects/corba.html.

[7] VanHeyningen, M., "The Unified Computer Science Technical Report Index: Lessons in indexing diverse resources," *2nd International World Wide Web Conference, WWW'94* Oct. 1994, pp. 535-543.

[8] Xerox Palo Alto Research Center, "Inter-Language Unification", URL: ftp://ftp.parc.xerox.com/pub/ilu/ilu.html

[9] Yan, T and H. Garcia-Molina, "SIFT – A Tool for Wide-Area Information Dissemination", *Proc. 1995 USENIX Technical Conference*, New Orleans, 1995, pp. 177-86. See also: http://sift.stanford.edu/