Digital Software and Data Repositories for Support of Scientific Computing*

Ronald Boisvert
National Institute of Standards and Technology

Shirley Browne†
University of Tennessee

Jack Dongarra
University of Tennessee and Oak Ridge National Lab

Eric Grosse
AT&T Bell Laboratories

Abstract

This paper discusses the special characteristics and needs of software repositories and describes how these needs have been met by some existing repositories. These repositories include Netlib, the National HPCC Software Exchange, and the GAMS Virtual Repository. We also describe some systems that provide on-line access to various types of scientific data. Finally, we outline a proposal for integrating software and data repositories into the world of digital document libraries, in particular CNRI's ARPA-sponsored Digital Library project.

*The work described in this paper was sponsored by NASA under Grant No. NAG 5-2736, by the National Science Foundation under Grant No. ASC-9103853, and by AT&T Bell Laboratories.

†Author to whom correspondence should be directed. 107 Ayres Hall, Computer Science Department, University of Tennessee, Knoxville, TN 37996-1301, (615) 974-5886, `browne@cs.utk.edu`

1 Introduction

Most work on digital libraries has focused on storage, retrieval, and display of digital forms of documents. A number of on-line software repositories have been developed that provide access to software and software artifacts. These document and software efforts have been mostly independent, with little attention paid to integrating the two types of libraries and to developing common principles for organization and operation.

This paper discusses the special characteristics and needs of software repositories and describes how these needs have been met by some existing repositories. These repositories include Netlib [20, 12], the National HPCC Software Exchange [13], and the GAMS Virtual Repository [8]. We also describe some systems that provide on-line access to various types of scientific data. Finally, we outline a proposal for integrating software and data repositories into the world of digital document libraries, in particular CNRI's ARPA-sponsored Digital Library project [28, 26].

2 Characteristics of Some Existing Software Repositories

2.1 Netlib

Netlib began services in 1985 to fill a need for cost-effective, timely distribution of high-quality mathematical software to the research community. Some of the libraries Netlib distributes – such as EISPACK, LINPACK, FFTPACK, and LAPACK – have long been used as important tools in scientific computation and are widely recognized to be of high quality. The Netlib collection also includes a large number of newer, less well-established codes. Most of the software is written in Fortran, but programs in other languages, such as C and C++, are also available.

Netlib sends, by return electronic mail, requested routines together with subsidiary routines and any requested documents or test programs supplied by the software authors [20]. Xnetlib, an interactive tool for software and document distribution [19], use an X Window interface and TCP/IP connections to allow users to receive replies to their requests within a matter of seconds. The interface provides a number of modes and searching mechanisms to facilitate searching through a large distributed collection of software and documents. World Wide Web browsers such as Mosaic and Netscape can also be used to access Netlib via HTTP and FTP [1].

Although the original focus of the Netlib repository was on mathematical software, the collection has grown to include other software (such as networking

[1]Netlib is accessible from a WWW browser at `http://www.netlib.org/`

tools and tools for visualization of multiprocessor performance data), technical reports and papers, a Whitepages Database, benchmark performance data, and information about conferences and meetings. The number of Netlib servers has grown from the original two, at Oak Ridge National Laboratory (initially at Argonne National Laboratory) and Bell Labs, to servers in Norway, the United Kingdom, Germany, Australia, Japan, and Taiwan. A mirroring mechanism keeps the repository contents at the different sites consistent on a daily basis, as well as automatically picking up new material from distributed editorial sites[24].

Netlib differs from other publicly available software distribution systems, such as Archie, in that the collection is moderated by an editorial board and the software contained in it is widely recognized to be of high quality. However, the Netlib repository is not intended to replace commercial software. Commercial software companies provide value-added services in the form of support. Although the Netlib collection is moderated, its software comes with no guarantee of reliability or support. Rather, the lack of bureaucratic, legal, and financial impediments encourages researchers to submit their codes by ensuring that their work will be made available quickly to a wide audience.

2.2 The National HPCC Software Exchange (NHSE)

The National HPCC Software Exchange (NHSE) is an Internet-accessible resource that will facilitate the exchange of software and information among research and computational scientists involved with High Performance Computing and Communications (HPCC) [13]. The purpose of the NHSE is to promote the development of discipline-oriented software and document repositories and of contributions to and use of such repositories by Grand Challenge teams, as well as by other members of the high performance computing community. The target audiences for the NHSE include HPCC application and computer scientists, users of government supercomputer centers, and potential industrial users. A prototype of the NHSE is accessible from a WWW browser at `http://www.netlib.org/nse/`.

The scope of the NHSE is software and software-related artifacts produced by and for the HPCC Program. Software-related artifacts include algorithms, specifications, designs, and software documentation. The following types of software are to be made available:

- Systems software and software tools. This category includes parallel processing tools such as parallel compilers, message-passing communication subsystems, and parallel monitors and debuggers.
- Data analysis and visualization tools.
- Basic building blocks for accomplishing common computational and communication tasks. These building blocks will be of high quality and trans-

portable across platforms. Building blocks are meant to be used by Grand Challenge teams and other researchers in implementing programs to solve computational problems. Use of high-quality transportable components will speed implementation, as well as increase the reliability of computed results.

- Research codes that have been developed to solve difficult computational problems. Many of these codes will have been developed to solve specific problems and thus will not be reusable as is. Rather, they will serve as proofs of concept and as models for developing general-purpose reusable software for solving broader classes of problems. The development of this reusable software is expected to be undertaken by commercial companies, rather than by academic researchers.

A catalog of the software currently available from the NHSE is accessible at `http://www.netlib.org/nse/sw_survey.html`.

Although the different disciplines will maintain their own software repositories, users should not need to access each of these repositories separately. Rather, the NHSE will provide a uniform interface to a virtual HPCC software repository which will be built on top of the distributed set of discipline-oriented repositories. The interface will assist the user in locating relevant resources and in retrieving these resources. A combined browse/search interface will allow the user to explore the various HPCC areas and become familiar with the available resources. A longer term goal of the NHSE is to provide users with domain-specific expert help in locating and understanding relevant resources.

2.3 GAMS Virtual Repository

The Guide to Available Mathematical Software (GAMS) project of the National Institute of Standards and Technology (NIST) studies techniques to provide scientists and engineers with improved access to reusable computer software components available to them for use in mathematical modeling and statistical analysis. One of the products of this work is the GAMS system, an on-line cross-index and virtual repository of mathematical software [8]. GAMS performs the function of an interrepository and interpackage cross-index, collecting and maintaining data about software available from external repositories and presenting it as a homogeneous whole. It also provides the functions of a repository itself (i.e., retrieval). However, instead of maintaining the cataloged software itself, it provides transparent on-demand access to repositories managed by others.

GAMS currently contains information on more than 9800 problem-solving software modules from about 85 packages found in four physically distributed software repositories (three maintained at NIST and Netlib). In addition to most of the software in the Netlib collection, GAMS cross indexes individual components in large multipurpose libraries such as IMSL, NAG, PORT, STARPAC and SLATEC, as well as capabilities of statistical analysis systems such as DATAPLOT and SAS. Both public-domain and commercial software is cataloged, and although source code of proprietary software products are not available through GAMS, related items such as documentation and example programs often are.

All problem-solving software modules in GAMS are assigned one or more problem classifications from a 736-node tree-structured taxonomy of mathematical and statistical problems developed as part of the project [9]. Users can browse through modules in any given problem class. To find an appropriate class, one can utilize the taxonomy as a decision tree, or enter keywords which are then mapped to problem classes. Search filters can be declared which allow users to specify preferences such computing precision or programming language. In addition, users can browse through all modules in a given package, or all modules with a given name. Each module's abstract lists the retrievable objects associated with the module, such as documentation, examples, test programs, source code and dependencies. (More than 32,000 such objects can be retrieved.)

At the core of the GAMS system is a relational database of information about available software. This database is maintained at NIST, which provides a classification service for the repositories it indexes. The GAMS network server provides this information to network clients using a specialized protocol over TCP/IP connections. In addition, a gateway to the World Wide Web has been developed using the Common Gateway Interface (CGI) facility of NCSA's httpd server [2].

3 Indexing and Searching of Software Objects

Cataloguing information for software objects serves two purposes – 1) to supply material from which a searchable index may be constructed, and 2) to supply information needed by the user to select/reject search hits and to obtain and use selected software. The field names and definitions used for cataloguing in a particular library are described in the *data model* used by that library. For example, for document libraries, the CSTR project [26] has adopted RFC1357 [14] as its data model.

3.1 Data Models

Data models used for document digital libraries are not in general suitable for use by software repositories. Although some fields are useful in both settings – e.g., author,title, abstract – software cataloguing requires a number of additional fields. A field that appears in most major catalogs is a **requirements** field that lists the hardware, operating system, and other software needed to use the

[2] Accessible at `http://gams.nist.gov/`

catalogued item. Another important field in the case of public domain software repositories such as Netlib, where software is author-supported, is a **contact** field giving an electronic mail address to which questions and bug reports may be sent. Still another field used by many software repositories is a certification field that tells at what level the software has been certified and possibly includes pointers to certification artifacts such as completed checklists and testing results.

The Reuse Library Interoperability Group (RIG) has developed and approved the Basic Interoperability Data Model (BIDM) as a standard data model for software repository interoperability, and the BIDM has been submitted for balloting as an IEEE standard [3]. The BIDM will be used as a lowest common denominator for interoperation between software repositories. However, because of considerable variation in the purpose, contents, and application domains of different software repositories, no single data model will be suitable for all, and important cataloguing information will be lost in exporting to the BIDM. An example where such loss occurs is the certification field. This field was not included in the BIDM because of the wide variety of certification and evaluation methods in use at different repositories. The RIG also encountered difficulty in developing controlled vocabulary lists, because again different sets of terms were appropriate for different repositories. The approach now being taken by the RIG is to define a standard for an Extensible Uniform Data Model (EUDM), which will be a meta-model a repository can use to describe the data model it is using. For example, using the EUDM, a repository will be able to define its certification methods and the meanings of different certification levels. As a member of the RIG, Netlib is participating in development and promotion of standard data models for software repositories.

3.2 Software Classification

A number of studies have shown that proper classification of software objects contributes to effective location of the objects by potential reusers [15, 31, 22]. Classification is carried out by assigning codes and/or keywords from a classification scheme or thesaurus. Classifications and thesauri developed for indexing documents, such as the INSPEC classification [1] and INSPEC thesaurus [2], are inadequate for indexing software objects. Firstly, these tools cover a broad range of topics and cover software-related topics only superficially. Thus, they do not allow the user to discriminate finely enough among the available software objects. Secondly, effective classification of software objects requires that the function of the object be indexed [10]. Because documents are not used as software is, terms related to function have not generally been included in thesauri developed for document indexing.

A classification scheme that has been developed specifically for use in indexing mathematical software is the GAMS hierarchy [9]. The GAMS scheme has been widely adopted by network-accessible repositories such as Netlib and by commercial mathematical software libraries such as NAG and IMSL. A successor to GAMS is currently under development. The new scheme will refine areas needing better discrimination and will add new categories to encompass recent developments in numerical algorithms. The new scheme will also be reorganized so as to be less cumbersome.

An HPCC software thesaurus is under development as part of the NHSE. This thesaurus uses GAMS categories for mathematical software but defines new terms for other areas of high performance computing. Some of these terms are drawn from an existing HPCC glossary [25] and from a book that gives an overview of parallel computing [21]. Terms from the INSPEC thesaurus [2] are being used for application areas. The thesaurus has a faceted structure with facets for application area, problem area, function, algorithm, and target architecture. The thesaurus will be made available on-line in hypertext form to assist with searching the NHSE, similar to [30]. Hypertext links from terms to scope and definition notes will also be provided.

3.3 Search Interfaces

Netlib currently provides a search interface [3] that allows the user to do field-specific keyword searching. For example, the user may search by author, filename, abstract keywords, or GAMS classification. Search results are returned as a hypertext list of catalog records from which the user may select files to view or download.

Searching by keywords or classification codes often returns a large number of search hits, leaving the user to sort through them. Further discrimination often cannot be provided by an overall classification scheme, but requires use of a domain specific knowledge base. Such knowledge bases have been constructed for specialized domains, including differential equations [29, 27, 4] and approximation [23]. We are experimenting with providing on-line hypertext interfaces to such knowledge bases. For example, we have provided a hypertext version of a decision tree for approximation algorithms [4].

We have also developed a prototype expert help system to assist users in selecting software within specific domains [7, 8]. An advisory system for a given problem class helps the user discriminate between problem-solving software modules for that class. For a given problem class, a set of features are partitioned into a small set of feature classes, and information is encoded about how each feature applies to each software module. Prototype user interfaces have been developed that allow the user to interact with choice widgets for the various features. The system provides more specific and effective help in selecting software than a domain-independent search interface.

[3] Accessible from a WWW browser at `http://www.netlib.org/nse/netlib_query.html`

[4] Accessible from a WWW browser at `http://www.netlib.org/a/catalog.html`

4 Retrieval of Software Objects

Once a user has located relevant software objects, he needs to be able to make use of them. Modes of use include the following:

1. Downloading, configuring, compiling, and executing a complete program or package.
2. Downloading routines and combining them with a user-supplied main program and other user-supplied routines before compiling and executing.
3. Using retrieved source code as a starting point for developing software with similar functionality.
4. Downloading templates or archetypes that provide a framework for writing actual code.
5. Using a remote execution service – i.e., shipping input data over a network to a remote execution server and then retrieving the resulting output data.

4.1 Downloading Files

A user may download files from Netlib by sending an email request or by clicking on the filenames from Xnetlib, GAMS, or WWW interfaces. Netlib files are organized into directories. Some directories contain a single package, such as the LAPACK directory, while others contain programs for a particular domain, such as the OPTimization directory. Each directory contains an index file containing catalog records for the files in the directory. Most directories also contain a readme file giving an overview of the directory. Some directories have subdirectories, for example for the different Fortran precisions available and for test and example programs. A user may initially do a keyword search to locate relevant directories and then browse the index files for those directories to locate relevant files which may then be downloaded.

When downloading a routine from Netlib, a user may make use of a dependency-checking mechanism that allows retrieval of the entire dependency tree for that routine. The user may specify that a subtree be omitted, however, if those routines have been retrieved previously. There is also an automatic tar facility that builds and returns a tar file of any Netlib directory or subdirectory upon user request. Binary executable files for several Netlib packages are also available – for example, for the Xnetlib and HeNCE packages.

Users may download files from the NHSE via a WWW browser. The NHSE provides a searchable catalog of HPCC software, as well as a browseable listing by category [5]. Each entry in the catalog includes either a URL that may be clicked on to retrieve the software or more information about it. Clicking on

[5] Accessible at `http://www.netlib.org/nse/sw_survey.html`

these URLs connects the user to the software provider's home site – i.e., the NHSE provides an interface to a virtual distributed repository consisting of a large number of independently maintained physical repositories. In the near future, the NHSE will switch from using URLs to using location independent names [11]. Use of location independent names will allow files to be moved without requires references to them to be changed and will permit transparent mirroring and reliable cacheing.

Care should always be exercised when downloading and using files obtained over a network, especially tar files and executables. Although Netlib is regarded by most users as a trusted source, it would be possible for someone to impersonate a Netlib server and make dangerous tar or executable files available, purportedly from Netlib. Because source code is unlikely to be examined closely by the user, deliberately introduced bugs or other malicious modifications might also slip past in source code form. In the near future, both Netlib and the NHSE will use public key cryptography to allow users to authenticate the source of downloaded files.

4.2 Templates and Archetypes

A template is a description of a general algorithm rather than the executable object code or the source code more commonly found in a conventional software library [5]. Templates may be customized by the user – for example, they can be configured for the specific data structure of a problem or for the specific computing system on which the problem is to run. Templates are written in a language-independent Algol-like structure, which is readily translatable into a target language such as Fortran or C. A collection of templates focusing on iterative methods for solving large sparse linear systems is available from Netlib [6]. For each template, the following is provided: a mathematical description of the flow of the iteration; discussion of convergence and stopping criteria; suggestions for applying a method to special matrix types (e.g., banded systems); advice for tuning (for example, which preconditioners are applicable and which are not); tips on parallel implementations; and hints as to when to use a method, and why.

A program archetype is (a) a program design strategy appropriate for a restricted class of problems, and (b) a collection of program designs with (c) implementations of exemplar problems in one or more programming languages and optimized for a collection of target machines. The program design strategy includes archetype-specific information about methods of deriving a program from a specification, methods of parallelizing sequential programs, the program structure, methods of reasoning about correctness and performance, empirical data on performance measurements and tuning for different kinds of machines, and suggestions for test suites. A project at Caltech is exploring the question

[6] Accessible at `http://www.netlib.org/linalg/html_templates/report.html`

of whether a library of parallel program archetypes be used to reduce the effort required to produce correct efficient programs [7].

4.3 Remote Execution

If a user needs to run a program only infrequently, and if compiling and installing the program involves considerable overhead, the user may prefer to take advantage of a remote execution service if one is available. Netlib has experimented with making remote execution of the Fortran-to-C converter (f2c) and Fortran checking (ftnchek) programs available. We configured the Mosaic WWW browser to invoke the Tcl language interpreter to execute downloaded files of type application/x-safe-tcl. We then made downloaded user interfaces for the f2c and ftnchek programs available on a Netlib server. Users could then download the interface modules and use them to interact with the remote execution services. The user interfaces allowed the user to select files to be transferred for processing and to set various options. Our work on remote execution is experimental at this stage because a safe client execution environment for Tcl has not yet been rigorously defined, although researchers at Sun Microsystems are working on it.

Software for using ORNL's GRAIL and GENQUEST remote execution services for doing DNA sequence analysis, gene assembly, and sequence comparison is available through the NHSE [8]. These services cannot be used from a WWW browser, but require downloading specialized X-Windows client software. In the future, the NHSE plans to support use of such services from a WWW browser by means of a safe execution environment for downloadable Safe-Tcl code, so that the client module may be executed directly from the browser.

Another possibility for remote execution is to allow users to upload executable code to a Netlib server and run it there. For example, the user might want to send an agent that would sift through and summarize computer performance data residing on Netlib. A search service such as Harvest might send an agent that would summarize the contents of Netlib and stream the summary back to an indexing engine. We are investigating the provision of this kind of user-directed remote execution using the Plan 9 operating system developed at AT&T Bell Labs.

4.4 Change Notification

Some digital document libraries have a notification service that informs subscribers of newly available documents. The notification service for a software repository is somewhat different, because it informs subscribers of changes and bug fixes to the software as much or more than of additions of new software.

---

[7] More information is available at `http://www.etext.caltech.edu/Papers2/ArchetypeOverview/ArchPaper.html`

[8] More information is available at `http://avalon.epm.ornl.gov/`

In the early days of the Netlib repository, when all access was by email and the traffic was mostly from professional numerical analysts, we relied on log files to send out notification of important bug fixes to everyone who had retrieved affected files. Now, because access is more anonymous and a wider spectrum of users are involved, the old scheme has been replaced by explicit subscription. People may indicate interest in specific Netlib directories, using *subscribe* and *unsubscribe* commands. Automatic notification is sent, on a daily basis, when files in the directory are changed. The subscriber lists also give the authors and editors a way to judge what community is particularly interested in a given Netlib collection.

5 Access to Scientific Data

The Netlib Performance Database provides an on-line catalog of public-domain computer benchmarks such as the Linpack Benchmark, Perfect Benchmarks, and the Genesis Benchmarks [6]. A benchmark code is a program designed to be run on an architecture so as to produce a relative measure of its execution. Benchmarks tend to evolve from individual applications that do not necessarily stress all features of a given architecture. Thus, benchmark numbers do not imply general machine performance but instead describe the performance of a machine on an algorithm or application class.

Although benchmarking has become very popular because of the diversity and competition in the computer hardware business, there was, previous to development of our database, no central repository for benchmark data. The WWW interface to our Performance Database [9] allows the user to

- view complete benchmark reports that disply sorted data from various published benchmark reports,
- browse the performance data tree by selecting the benchmark and machines about which information is desired,
- search the performance database

There are also pointers to benchmark papers and other benchmark and performance-related literature.

Various archives of scientific data are accessible from the NHSE – for example, NASA's Planetary Data System [10] and Astrophysics Data System [11], NIST's Atomic Spectroscopic Database [12], and NOAA's Environmental Data Centers [13]. There is no uniform cataloguing method or search interface for these

[9] Accessible at `http://performance.netlib.org/performance/html/PDStop.html`
[10] Accessible at `http://stardust.jpl.nasa.gov/pds_home.html`
[11] Accessible at `http://adswww.harvard.edu`
[12] Accessible at `http://aeldata.phy.nist.gov/nist_beta.html`
[13] Accessible at `http://www.esdim.noaa.gov/`

databases, nor a standard way of describing the contents and services offered. Thus, the user has no way of systematically discovering relevant databases and must learn a different interface for each one.

6 Integration with Document Digital Libraries

The Corporation for National Research Initiatives (CNRI) is working with five major universities (CMU, Cornell, UC-Berkeley, Stanford, and MIT) on an ARPA-sponsored project to develop concepts for digital libraries. As part of this project, each university is placing its Computer Science Technical Reports on-line and providing access to the distributed CSTR collection. Technologies developed for the CSTR project include the Dienst distributed search system [18] and a Handle Management Service for assigning, maintaining, and using unique identifiers for digital library objects [16].

The basic architecture being developed by CNRI for distributed digital libraries includes the following concepts [26]:

- A *digital object* which consists of a sequence of bits plus a unique identifier known as a *handle* (the binding between the handle and the sequence of bits may change over time).
- *Naming authorities* who are responsible for assigning unique identifiers within their portions of the handle namespace.
- *Repositories* from which digital objects are available.
- *Information and Reference (IR) servers* that provide reference information about collections of digital objects.

The CNRI work is closely related to the IETF Uniform Resource Identifier (URI) Working Group's work on Uniform Resource Names (URNs) [32] and Uniform Resource Citations [17]. CNRI's handle is the equivalent of IETF's URN, and CNRI's IR server serves a similar function to IETF's URC server.

The Netlib and NHSE Development Group has been engaged in a parallel effort to implement a location-independent naming architecture [11]. We provide for two types of location-independent names:

- a Uniform Resource Name (URN), for which the contents it refers to may change – e.g., the "current version of LAPACK".
- a Location Independent File Name (LIFN), for which the binding between the name and the byte contents of the file it refers to is fixed, once assigned. This type of name is needed for unambiguous references when attaching critical reviews or reporting scientific results obtained using a particular version of a piece of software. LIFNs also permit reliable and efficient cacheing and mirroring of files.

At any given time, a URN is associated with exactly one LIFN. By looking up the LIFN associated with a URN and then retrieving the file corresponding to that LIFN, the user is assured of retrieving the most recent copy, even if some mirrored copies are out-of-date. Thus, we obtain consistency of replicated copies without the overhead of a replica control protocol.

We are also developing a URC server that provides support for the following:

- Provision by the publisher of attribute-value pairs for a given URN in the form of cryptographically signed assertions.
- Retrieval and authentication of assertions by users.
- Specification of the data model used for a particular URN.
- Choice of encryption algorithm, including none.

We propose to integrate our software repository naming architecture with CNRI's digital library architecture in the following manner:

1. Both URNs and LIFNs will be expressible as handles, and URN and LIFN lookup will be merged with the Handle Management Service.
2. Our URC server will be an implementation of CNRI's IR server that may be used for cataloguing general Web resources, including software and data archives.
3. Similar to the Dienst protocol for document repositories, we will develop service specifications and retrieval protocols appropriate for software and data repositories. In addition, similar to the Digital Library Document Architecture that defines requirements for digital document structure [33], we will define requirements for software and data archive structures.

References

[1] *INSPEC Classification.* Institution of Electrical Engineers, 1992.

[2] *INSPEC Thesaurus.* Institution of Electrical Engineers, 1993.

[3] Standard reuse library Basic Data Interoperability Model (BIDM). Technical Report RPS-0001, Reuse Library Interoperability Group, 1993.

[4] C. A. Addison, W. H. Enright, P. W. Gaffney, I. Gladwell, and P. M. Hanson. Algorithm 687: A decision tree for the numerical solution of initial value ordinary differential equations. *ACM Trans. Math. Softw.*, 17(1):1–11, Mar. 1991.

[5] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Dunato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 1994.

[6] M. W. Berry, J. J. Dongarra, B. H. Larose, and T. Letsche. PDS: A performance database server. *Scientific Computing*, 3(2):147–156, 1994.

[7] R. F. Boisvert. Toward an intelligent system for mathematical software selection. In P. W. Gaffney and E. N. Houstis, editors, *Programming Environments for High-Level Scientific Problem Solving*, pages 79–92. North-Holland, Amsterdam, 1992.

[8] R. F. Boisvert. The architecture of an intelligent virtual mathematical software repository system. *Math. & Comp. in Simul.*, 36:269–279, 1994.

[9] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. The Guide to Available Mathematical Software problem classification system. *Comm. Stat. – Simul. Comp.*, 20(4):811–842, 1991.

[10] E. J. Breton. Indexing for invention. *Journal of the American Society for Information Science*, 42(3):173–177, 1991.

[11] S. Browne, J. Dongarra, S. Green, K. Moore, T. Pepin, T. Rowan, and R. Wade. Location-independent naming for virtual distributed software repositories. In *ACM-SIGSOFT Symposium on Software Reusability*, Apr. 1995. (to appear).

[12] S. Browne, J. Dongarra, S. Green, K. Moore, T. Rowan, and R. Wade. Netlib services and resources. Technical Report UT-CS-94-222, University of Tennessee Computer Science Department, Feb. 1994.

[13] S. Browne, J. Dongarra, S. Green, K. Moore, T. Rowan, R. Wade, G. Fox, K. Hawick, K. Kennedy, J. Pool, and R. Stevens. The national HPCC software exchange. *IEEE Computational Science and Engineering*, 1995. (to appear).

[14] D. Cohen. A format for e-mailing bibliographic records. Internet RFC 1357, July 1992.

[15] P. Constantopoulos and E. Pataki. A browser for software reuse. In *Advanced Information Systems Engineering, 4th International Conference CAiSE '92, Porceedings*, pages 304–326. Springer-Verlag, Berlin, Germany, May 12–15 1992.

[16] Corporation for National Research Initiatives. Interfacing to the handle management system. Accessible at http://www.cnri.reston.va.us/home/cstr/handle-intro.html, 1994.

[17] R. Daniel Jr. and M. Mealling. URC scenarios and requirements. Internet Draft draft-ietf-uri-urc-req-00.txt, Nov. 1994.

[18] J. R. Davis and C. Lagoze. Dienst, a protocol for a distributed digital document library. Internet Draft accessible at http://cs-tr.cs.cornell.edu/Info/dienst_protocol.html, July 1994.

[19] J. Dongarra, T. Rowan, and R. Wade. Software distribution using XNETLIB. *ACM Trans. Math. Softw.*, 21(1), 1995 1995.

[20] J. J. Dongarra and E. Grosse. Distribution of mathematical software via electronic mail. *Commun. ACM*, 30(5):403–407, May 1987.

[21] G. Fox, R. D. Williams, and P. Messina. *Parallel Computing Works*. Morgan Kaufmann, 1994.

[22] W. B. Frakes and T. P. Pole. An empirical study of representation methods for reusable software components. *IEEE Trans. on Software Engineering*, 20(8):617–630, Aug. 1994.

[23] E. Grosse. A catalogue of algorithms for approximation. In J. Mason and M. Cox, editors, *Algorithms for Approximation II*, pages 479–514 (of 514), London, England, 1990. Chapman and Halll.

[24] E. Grosse. Repository mirroring. *ACM Trans. Math. Softw.*, 21(1), Mar. 1995.

[25] K. A. Hawick. High Performance Computing and Communications glossary. Technical report, Northeast Parallel Architectures Center at Syracuse University, July 1994.

[26] R. Kahn and R. Wilensky. Locating electronic library services and objects: A frame of reference for the CS-TR project. Draft for discussion purposes, Feb. 1994.

[27] M. S. Kamel, K. S. Ma, and W. H. Enright. ODEXPERT: An expert system to select numerical solvers for initial value ODE systems. *ACM Trans. Math. Softw.*, 19(1):44–62, Mar. 1993.

[28] R. R. Larson. Design and development of a network-based electronic library. In *Proc. ASIS Mid-Year Meeting*, pages 95–114, Portland, Oregon, May 1994.

[29] M. Lucks and I. Gladwell. Automated selection of mathematical software. *ACM Trans. Math. Softw.*, 18(1):11–54, Mar. 1992.

[30] R. Pollard. A hypertext-based thesaurus as a subject browsing aid for bibliographic databases. *Information Processing and Management*, 29(3):345–357, 1993.

[31] R. Prieto-Diaz. Implementing faceted classification for software reuse. *Commun. ACM*, 34(5):88–97, May 1991.

[32] K. Sollins and L. Masinter. Functional requirements for uniform resource names. Internet RFC 1737, Dec. 1994.

[33] W. Turner. The document architecture for the Cornell Digital Library. Internet RFC 1691, Aug. 1994.