The National HPCC Software Exchange Repository Planning Guide

Shirley Browne
University of Tennessee
browne@cs.utk.edu

June 12, 1996

Contents

1 Purpose of the Guide

The purpose of this document is to give guidelines for the development of domain-specific HPCC repositories that will provide access to software and documents within their specific domains, as well as interoperate with each other and with the top-level NHSE interface. Specific guidelines are given for scoping the domain, classifying and cataloging software, access mechanisms and control, securing the repository site(s), interoperation, measurement, and ongoing maintenance and development. Guidelines for developing and utilizing a domain-specific software review framework that extends the top-level NHSE software review framework are contained in a separate document entitled the *NHSE Software Review Framework*.

A toolkit called Repository in a Box (RIB) is under development by the NHSE, and the first version is expected to be released in August 1996. This version will contain tools for the following:

- creating and maintaining software catalog records,
- exchanging catalog records with other repositories, including export to the NHSE virtual repository,
- mirroring files from other sites,
- setting up browsing and searching interfaces to the repository,
- usage statistics collection, analysis, and reporting.

The aim of the NHSE is to provide a single entry point to a distributed collection of domain-specific HPCC repositories. Domain experts are the best qualified to select, evaluate, catalog, and provide access to software within their specialized domains. The intended audience of the NHSE is the HPCC user community, consisting of Grand Challenge research teams, other government and academic researchers, users of supercomputer centers, and industrial users. Software sharing within the NHSE is intended to be both intradisciplinary and interdisciplinary. The expected benefits of the NHSE are reduction of duplication of time and effort in software development and faster production of higher quality software.

2 Scoping the Domain

To be most effective, a domain-specific repository should be a well-organized collection of high-quality software assets with well-defined scopes of applicability. An *asset* is any reusable product of the software development life cycle – e.g., software component, template, design, specification, test suite. The purpose of scoping the domain is to determine the domain's definition, boundaries, relationships with other domains, and potential customers. The scoping task results in a set of objectives for the repository, as well as criteria for selecting which assets should be included in the repository.

Scoping the domain should be carried out within the domain's *organizational context*, which consists of the stakeholders, funding and management structure, and team background and culture for projects within the domain. Input is needed from stakeholders as their interest, needs, and criteria for success of the project.

To carry out the scoping task, repository developers should determine answers to the following questions:

(An example set of answers to these questions that resulted from the scoping task for HPC-netlib, the high performance branch of the Netlib mathematical software repository, is included in Appendix A).

1. What are the specific desired outcomes/goals of this repository?
2. What classification schemes already exist for classifying software in this domain? Are they sufficient? If not, how should a classification scheme be developed?
3. How is this domain interrelated with other domains – e.g., parallel systems software, math software, applications domains?
4. What are important bibliographies, survey articles, and literature in this domain?
5. Which software packages in this domain are popular or considered to be of high quality? What are the distinguishing features or characteristics that account for their success?
6. What software architectures characterize software systems in the domain (Shaw, Abstractions for Software Architectures and Tools to Support Them, IEEE Trans. on Soft. Eng., 1995)? The type of software architecture affects whether reuse in a domain will be small-scale reuse of single components or large-scale reuse of aggregates of components. If architectures are very special-purpose, software may not be reusable, but other artifacts (e.g., algorithms, requirements specifications) might be.
7. What are the current mechanisms, conventions, and avenues for software distribution and technology transfer (i.e., to industry) in this domain?

8. Who are specific expected customers of this repository and what are their needs?

9. Who is funding the repository project and what are their expectations and criteria for success?

Following the above question answering exercise, the repository developers should produce the following items and obtain agreement on them from the various stakeholders, including funders, asset producers, and anticipated customers:

- A concise statement of the goals and objectives of the repository project.
- A concise definition of the subject area covered by the domain, together with a set of criteria for determining what software falls within the domain and some examples of software systems that do and do not fall within the domain.
- A selected taxonomic classification scheme or a plan for producing such a scheme.
- A list of potential customers and their anticipated requirements.
- A small number of exemplary software packages to be included in the repository initially, together with their anticipated uses and specific customers.

3 Classification and Cataloging

Information about the software in a repository should be systematically organized into a catalog that may be efficiently browsed and searched. Having a software librarian put some effort into manually classifying and cataloging the software will enable more precise searching by end users.

Software is classified by labelling it with one or more descriptors from a classification scheme. Possibilities for the classification scheme include a hierarchical taxonomy [4], a keyword thesaurus [2], a faceted thesaurus [6], or a combination of these. A hierarchical taxonomy usually classifies according to a single characteristic, such as problem area for the GAMS mathematical software taxonomy [4]. A full-featured thesaurus is a collection of terms with broader-term, narrower-term, related-term, and use(d)-for relationships. The INSPEC thesaurus is an example of a keyword thesaurus used to classify the scientific and engineering research literature. A faceted thesaurus organizes its collection of terms into characteristic groups called facets. Possible facets for software indexing terms might be problem area, application area, and algorithm/method. A faceted thesaurus may be thought of as consisting of multiple hierarchical taxonomies (i.e., the broader-term/narrower-term relationships within each facet), but with the possibility of relationships between terms in different facets.

Some repository developers will be fortunate enough to find existing classification schemes for their domains which they can use as is or easily adapt. Others will be faced with the task of creating a new scheme or carrying out non-trivial extension of an existing one. Except for very small or specialized domains, development of a high-quality classification scheme will be a large-scale effort requiring the collaboration of experts in various specialties. Classification schemes also require ongoing maintenance, because new terms must be added when new items do not fit existing categories or vocabulary.

Factors to be considered in selecting or developing a classification scheme include the following:

- Anticipated size and rate of growth of the database to be indexed

  A large database requires the use of precision devices, such as decision trees [1] or expert advisory systems [5, 3], to effectively discriminate between large numbers of similar items.

- Expertise level and quantity of available manpower for doing the classification and cataloging

  Having authors assign controlled vacabulary terms to their own works has been shown to produce problems with indexing consistency [2].

- Expertise of end users of the software catalog

  Inexpert users can be assisted by including explanatory notes for categories and vocabulary terms and by providing automatic translation of natural language to controlled language.

Catalog records for software contain values the the attributes of the cataloged software and for the relationships between software components and between software and other entities such as authors, repositories, and individual files. The attributes and relationships are best expressed in terms of an abstract data model which can then be mapped to a concrete syntax for storage, retrieval, and exchange with other repositories.

To facilitate interoperation between participating HPCC repositories, the NHSE recommends use of the Reuse Library Interoperability Group (RIG) Basic Interoperability Data Model (BIDM) for exchanging software catalog records between repositories. The BIDM is an IEEE standard (1420.1) that specifies a minimal set of catalog information that a software repository should provide about its software in order to interoperate with other repositories. The BIDM is expressed in terms of an extended entity-relationship data model that defines classes for assets (the software entities), the individual elements making up assets (i.e., files), reuse libraries (i.e., repositories) that provide assets, and organizations that develop and manage libraries and assets. The model was derived from careful study and negotiation of the commonalities between existing academic, government, and commercial reuse libraries, by representatives from these libraries. Reuse libraries need not adopt the BIDM internally, although many have. They can continue to use internal search and classification mechanisms appropriate to their unique missions while using the BIDM as a uniform external interface.

The RIG has developed an extension to the BIDM called the Asset Certification Framework (ACF), which is the basis of the NHSE Software Review Framework, which is described in a different document. The ACF provides classes, attributes, and relationships which may be used to express a reuse library's certification policy and certification results so that they may be easily interpreted by other libraries. The NHSE recommends that domain-specific HPCC repositories use the RIG ACF for describing their software review policies and for expressing results of software review efforts.

Another BIDM extension currently under development and which the NHSE expects to adopt in the near future is the Intellectual Property Rights Framework (IPRF) for describing software legal restrictions, such as copyright, patents, licenses, and export restrictions.

The classification schemes discussed above currently fit into the *keyword* asset attribute of the BIDM. There is currently no way to specify from what scheme the values for the keyword attribute are drawn. However, the RIG is currently working on a meta-model that address the controlled vocabulary problem and that will also allow for domain-specific extensions of the BIDM.

The RIG has developed two bindings of the BIDM and the ACF which map the abstract data model to concrete syntax specifications that can be used for exchange of asset catalog information via the World Wide Web. The Web bindings, one that maps the BIDM to an SGML Document Type Definition (DTD), and another that maps to META and LINK tags in the header of an

HTML document, have been defined and are currently being implemented and tested.

More information about the RIG, the BIDM, and the Web bindings is available from the RIG home page at `http://www.rig.org/`.

The NHSE is developing a Repository in a Box (RIB) tookit that will assist repository developers in creating and maintaining software catalog records, in exchanging these records with other repositories (including the top-level virtual NHSE repository), and in providing a user interface for the software catalog. Specifically, RIB will include the following:

- a Web form and cgi-bin program that enables authors and librarians ito enter catalog information and produces records in the form of the RIG Web bindings,
- a Web spider that gathers BIDM records from other sites for inclusion in the local catalog, and
- a browse/search interface that can incorporate domain-specific classification schemes.

RIB is expected to be available by August of 1996.

For more details about repository interoperability, see section 6.

4 Access Mechanisms and Control

The NHSE anticipates that most HPCC repositories will provide access to software and documents by running an http server, as well as perhaps an ftp server. Users will most likely access these servers from a WWW browser, such as Netscape or NCSA Mosaic. Given the URL for a file the user wishes to retrieve, the browser will send the necessary request to the server and either display the retrieved file in a window or save it to the user's disk.

There is not currently a good way for Web browsers to retrieve multiple files in one request – for example, all files making up a software package – unless the files have been pre-packaged into a single tar or shar file. It would be convenient for a user to use a shopping basket approach to specify an arbitrary collection of files and then download them in the background. NHSE developers have been participating in discussions within the WWW community on incorporating such bulk file transfer into WWW protocols such as HTTP.

An alternative to making software available for download is to provide a remote execution service, with which users can submit their input data to a remote server that runs the program and returns the results. Examples of this approach include the NEOS optimization server [1] and the Web interface to //ELLPACK [2]. If there is sufficient demand for them, tools for setting up and running such remote execution services may be included in future versions of RIB.

Some software packages will have access restrictions which require authentication of authorized users before the software can be downloaded. Access control mechanisms available with http client-server combinations include passwords, prohibition by domain name, prohibition by IP address, configurable user groups, rules based on URLs, and the SSL and S-HTTP protocols. A chart showing which http servers support which features may be found at `http://www.webcompare.com/server-main.html`. Good sources of information about access control include the WWW Security FAQ [3] and the NCSA httpd documentation [4].

An alternative to restricting access to the actual software files is to encrypt these files and allow anyone to download them, but require authorization and authentication to obtain the decryption key. This approach is used in the Crytolope technology of IBM Infomarket [5]. A similar approach is under consideration by the NHSE for distributing software that has access restrictions.

---

[1] `http://www.mcs.anl.gov/home/otc/Server`

[2] `http://www.cs.purdue.edu/research/cse/pellpack/pellpack.html`

[3] `http://www.genome.mit.edu/WWW/faqs/www-security-faq.html`

[4] `http://hoohoo.ncsa.uiuc.edu/security/`

[5] `http://www.infomkt.ibm.com/`

5 Securing the Repository Site(s)

Repository maintainers should take reasonable precautions to protect their server machine(s) from breakins. Good sources of information on Web site security include the WWW Security FAQ [6] and the NCSA httpd documentation [7].

Additional security precautions include the following:

1. limiting services on the server machine to those that are absolutely necessary,
2. limiting logins on the server machine,
3. using a secure login protocol such as S-Key for the server machine.

[6] http://www.genome.mit.edu/WWW/faqs/www-security-faq.html

[7] http://hoohoo.ncsa.uiuc.edu/security/

6 Repository Interoperation

The motivation for repository interoperation is to facilitate inter-organization and inter-disciplinary software sharing. A high quality repository for one domain may well contain software that is useful in other domains. For example, software from the mathematical software and parallel tools repositories will be useful across a number of application domains. Through interoperation maintainers of domain-specific repositories will be able to import selected software from other repositories and incorporate it into their own catalogs, perhaps with value-added information about how the software is used within their domain. Furthermore, different organization that maintain repositories within the same domain will be able to share software. The top-level NHSE directory will import software catalog records from all the participating domain-specific repositories in order to provide a single index to all the available HPCC software, and will thus increase the visibility of this software.

Software may be shared between interoperating repositories at two levels:

1. at the level of catalog information which describes the software,
2. at the level of the actual software files.

To carry out interoperation at the first level, the NHSE recommends the exchange of RIG BIDM catalog records using one of the Web bindings, as described in section 3.

A repository that interoperates at the second level would mirror the files for imported software and distribute this software to users itself, rather than having users retrieve the software from the owning repository. Advantages of the mirroring approach may be the provision of faster and more reliable service to users, and a single point of contact for users for administrative procedures such as licensing agreements. Problems with the mirroring approach are liability for enforcing legal restrictions and proper crediting of download and usage statistics to the originating repository.

The Repository in a Box (RIB) toolkit will facilitate interoperability by providing the following:

1. tools for the first level of interoperability, as described in section 3,
2. mirroring software,
3. usage statistics collection and reporting software.

The enforcement of legal restrictions, including liability issues, is currently under study by the NHSE. Once these issues have been resolved, mechanisms for enforcing legal restrictions may possibly be included as part of RIB.

The capabilities of assigning a location-independent name, such as a Uniform Resource Name (URN), to a software resource, of easily attaching catalog information from independent sources (e.g., author, librarian, reviewer) to such a name, and of resolving a name to one or more identical replicated copies, would greatly facilitate repository interoperation. The NHSE is monitoring progress on these issues, and RIB may in the future incorporate mechanisms for assigning and resolving location-independent names.

7 Measurement

Repository maintainers need to collect, analyze, and report usage statistics for several reasons:

- to justify their existence to their funders,
- to provide feedback to authors on the extent and nature of uses of their software,
- for advertising purposes, so as to be considered a popular Web site and thus attract even more users,
- to deal with server bottlenecks and overload conditions.

A server program normally writes access records to a log file. In the case of http, these records contain only the domain name or IP address of the requesting machine, not the identity of the individual user. In the case of anonymous ftp, the log records contains the email address typed in by the user. It is possible for an http server to provide access via a cgi-bin program that requests information from the user, such as name and email address. Many users would consider it an annoyance to see the request on every file download. If a mechanism for bulk file download is adopted by the WWW community, it will be possible for the user to supply the information once per bulk download.

Tools to collect, analyze, display, and report log statistics (including reporting to home sites of mirrored files) will be available as part of the NHSE's Repository in a Box (RIB).

Download counts only tell how many times a file has been requested. They tell nothing about whether or how the software has actually been used. To obtain this important information, repository maintainers will need to try to contact users who have downloaded the software. This followup might most practically be done on a per-package basis by the software developers, perhaps through a user group mailing list. Repository maintainers might provide assistance to authors in carrying out followup surveys.

8 Ongoing Maintenance, Development, and User Support

Repository maintainers will need to monitor server logs and make needed changes to keep services running smoothly. They will want to keep abreast of new releases of server software and update their servers as necessary to take advantage of new features and bug fixes. Web pages should be maintained and checked for broken links using a tool such as MomSpider [8].

Authors of repository software need an easy-to-use method to notify repository maintainers of updates to their software, and to have new versions uploaded to the repository.

When a repository is just getting started, its initial collection should be a small number of software packages that are known to be of high quality and usefulness and that are targeted toward specific customers. Based on feedback from these customers, and using selection criteria determined during the domain scoping task described in section 2, the repository can gradually acquire additional software. If the repository is of high quality, contributors will come knocking at its door.

Repository maintainers need to make a decision about the level of user support they will provide. Options range from

1. no support whatsoever, as on some ftp sites, to
2. answering content-independent questions such as how to download, unpack, and compile files, but referring questions about specific software to the authors, to
3. offering a minimal level of support for the software but referring more complicated questions to the authors, to
4. fully supporting the software.

An approach to user support that works well for the Netlib mathematical software repository [9] is to maintain a Frequently Asked Questions file to try to handle the most common questions immediately and to provide an email hotline for questions not found in the FAQ. The email hotline is manned by several Netlib maintainers with different areas of expertise. The Netlib maintainers answer what questions they are able to and refer questions to authors when appropriate. In some cases, where the user appears to have a local problem or needs considerable hand-holding, he is encouraged to seek the assistance of a local systems administrator or numerical consultant.

[8] `http://www.ics.uci.edu/WebSoft/MOMspider/`

[9] `http://www.netlib.org/`

References

[1] 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.

[2] J. Aitchison and A. Gilchrist. *Thesaurus construction: a practical manual.* Aslib, 1987. ISBN 0-85142-197-0.

[3] 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.

[4] 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.

[5] 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.

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

Appendix A. Answers to Scoping Questions for HPC-netlib

HPC-netlib: High Performance Math Software Repository

1. Desired outcomes/goals
   (a) Enable easy access to mathematical software designed for vector and parallel supercomputers
   (b) Assist non-math users in selecting and using high performance math software
   (c) Develop and begin using evaluation framework for high performance math software
   (d) Target and meet needs of specific HPCC agency customers, such as DoD HPCMP, NASA Center for Computational Sciences, DOE and NSF supercomputer centers.
2. Classification schemes

   The GAMS taxonomy for classifying mathematical software already exists. GAMS is a single-faceted classification scheme which differentiates by problem area only. Additional facets, such as algorithm/method and programming model/machine architecture, may be necessary for classifying high performance math software.
3. Relationships with other domains
   (a) Numerical libraries for parallel machines perform communication through interface to underlying communications library – e.g., PVM, MPI, or vendor-specific.
   (b) Many scientific and engineering applications rely on a small core of linear algebra routines.
   (c) Physical simulations in a number of disciplines involve solving systems of PDEs.
   (d) Large dense linear systems arise from the solution of boundary integral equations in electromagnetics (radar cross section problem - Helmholtz equation) and fluid mechanics (Laplace equation), and from quantum mechanical scattering (numerical solution of either the time dependent or time independent Schrodinger equations, or of the Lippmann-Schwinger equation). [Edelman, 1993]
   (e) Computational chemists solving some form of Schrodinger's equation are the main generators of large dense eigenvalue problems (real symmetric or complex Hermitian. [Edelman, 1993]

4. Important literature

   Linear algebra:

   Gallivan, Heath, Ng, Ortega, Peyton, Plemmons, Romine, Sameh, and Voigt, Parallel Algorithms for Matrix Computations, SIAM, 1990.

   Demmel, Heath, and Van der Vorst, Parallel Numerical Linear Algebra, Acta Numerica 1993 (also /netlib/lapack/lawns/lawn60.ps).

   Optimization:

   Optimization Software Guide by Jorge J. More' and Stephen J. Wright

   Jorge J. More' and Brett M. Averick, Evaluation of Large-Scale Optimization Problems on Vector and Parallel Architectures, SIAM Journal on Optimization 4:4 (nov 1994), pp. 708-721.

   PDEs:

   Johnson, Numerical solution of partial differential equations by the finite element method, Cambridge University Press, 1987

   Johnsson and Mathur, Data Structures and Algorithms for the Finite Element Method on a Data Parallel Supercomputer, Int. J. Num. Meth. Eng. 29, 1990

   Chrisochoides, Houstis, and Rice. Mapping Algorithms and Software Environments for Data Parallel PDE Iterative Solvers. Journal of Parallel and Distributed Computing 21:75-95, 1994.

   FFTs:

   Swarztrauber, Multiprocessor FFTs, Parallel Computing 5 (1987), pp. 197-210

   Briggs, Hart, Sweet, and O'Gallagher, Multiprocessor FFT Methods, SIAM J. Sci. Statist. Comput., 8, pp. 27-42.

5. Popular, high-quality software packages

   (a) LAPACK and ScaLAPACK
   (b) BLAS and PBLAS
   (c) BLOCKSOLVE95
   (d) PCG
   (e) SVDPACK
   (f) //ELLPACK
   (g) Petsc
   (h) MINPACK-2

6. Software architecture

7. Distribution and technology transfer

    (a) Distribution from Netlib, discussion on NA-Digest.
    (b) BLAS, LAPACK, and ScaLAPACK adoption by industry are examples of successful technology transfer.

8. Expected customers:

    (a) NASA Center for Computational Sciences (NCCS)
        http://sdcd.gsfc.nasa.gov/NCCS/
        Provides computing resources and support services for Earth and space science researchers.
    (b) NSF National MetaCenter for Computational Science and Engineering
        http://www.ncsa.uiuc.edu/General/MetaCenter/MetaCenterHome.html
        Offers high performance and scalable parallel architectures for scientists and researchers to attack Grand Challenge problems.
    (c) DOD High Performance Computing Modernization Program
        http://www.hpcm.dren.net/
        Common HPC Software Support Initiative (CHSSI)
        Major Shared Resource Centers (MSRCs)
    (d) DOE National Energy Research Supercomputer Center
        http://www.nersc.gov/
        Principal supplier of production high-performance computing and networking services to the nationwide energy research community.
    (e) HPC providers wishing to satisfy S&ECWG and NCO task force Baseline Development Environment and Priority Capabilities
        http://www.nero.net/ pancake/SSTguidelines/
        Section 3.4 Math Libraries

9. Funder expectations and success criteria