Software Repository Interoperability

Shirley Browne
Jack Dongarra
University of Tennessee

Kay Hohn
ASSET

Tim Niesen
Raytheon

Technical Report UT-CS-96-329

July 1996

Abstract. A number of academic, commercial, and government software repositories currently exist that provide access to software packages, reusable software components, and related documents, either via the Internet or via intra-organizational intranets. It is highly desirable, both for user convenience and savings in duplication of effort, that these repositories interoperate. This paper describes interoperability standards that have already been developed as well as those under development by the Reuse Library Interoperability Group (RIG). These standards include a data model for a common semantics for describing software resources, as well as frameworks for describing software certification policies and intellectual property rights. The National HPCC Software Exchange (NHSE) is described as an example of an organization that is achieving interoperation between government and academic HPCC software repositories, in part through adoption of RIG standards.

1 Introduction

Over the past decade, a number of academic, commercial, and government software reuse libraries, also called software repositories, have come into existence. Examples include the Netlib mathematical software repository, ASSET, ELSA/MountainNet, the Army Reuse Center, and the Raytheon Corporation Reuse Library. These libraries generally contain software packages and reusable software components, as well as relevant documents such as technical reports and standards documents. Some of the libraries, such as Netlib, are domain-specific, while others contain software from a range of disciplines. Some of the software is freely distributed, while some requires license agreements and/or payment.

A software repository is more than just a warehouse of software files. Staff persons typically classify and catalog the software, carry out evaluation and certification procedures, and provide some level of support to users. A software catalog is usually maintained which may be browsed or searched by users.

Although the existence of many independent software repositories is desirable because it allows each repository to tailor its contents and services to a particular application domain or community of users, multiple independent repositories can also result in redundant and inefficiency. It is inconvenient for the user to access each repository separately. It would be preferable for software libraries to interoperate so that a user of any one library could obtain goods and services offered by other libraries, and so that libraries would not have to duplicate holdings in order to offer comprehensive services to their users.

Recognizing the above mentioned advantages of interoperability, a group of corporations and government reuse library programs founded the Reuse Library Interoperability Group (RIG) in 1991. The current RIG membership consists of over twenty organizations, including government and academic reuse libraries and corporations. Online information about the RIG may be found at the URL `http://www.rig.org/`. The purpose of the RIG is to develop standards for reuse library interoperability. Since 1994, the RIG has worked with the formal standardization process of the Software Engineering Standards Committee (SESC) of the IEEE. The RIG had its first IEEE standard, 1420.1, the Basic Interoperability Data Model (BIDM), approved in December of 1995.

Software repositories may interoperate at the following two levels:

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

2. at the level of the actual software files.

Interoperation at the first level requires a common semantics for catalog records, and an agreed upon common syntax in which to exchange these records. Although the user would browse and search for software from all the interoperating repositories from a single interface, he might still need to retrieve the actual software files from the owning repository.

A repository that interoperates at the second level would mirror the files for imported software and distribute them 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 include ensuring proper execution of payment and of licensing agreements, liability for enforcing legal restrictions, and proper crediting of download and usage statistics to the owning repository.

The remainder of this chapter is organized as follows. Section 2 describes the RIG Basic Interoperability Data Model, which specifies a common semantics for describing software resources. Section 3 describes bindings of the BIDM to concrete syntax for exchange and experiences with using these bindings. Section 4 discusses extensions to the BIDM to handle descriptions of software certification policies and results and of intellectual property rights and other legal restrictions, as well as preliminary work on a formal model for carrying out such extensions. Section 5 gives an overview of how the National HPCC Software Exchange is achieving interoperability among government and academic HPCC software repositories, in part through adoption of RIG standards.

2 The Basic Interoperability Data Model

The Basic Interoperability Data Model (BIDM), which is an IEEE standard (1420.1) for software reuse libraries, specifies a minimal set of metadata that a reuse library should provide about its reusable assets in order to interoperate with other reuse libraries [1]. The BIDM is expressed in terms of an extended entity-relationship data model that defines classes for assets (the reusable entities), the individual elements making up assets (i.e., files), libraries 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 BIDM may be visualized using the graphic notation of James Rumbaugh's *Object-Oriented Modeling and Design* [7]. Figure 1 provides a legend for the graphic notation. A pictorial view of the BIDM is shown in Figure 2.

A subclass inherits all attributes and relationships of its parent class. For example, the Asset, Element, Library, and Organization classes all inherit the Name attribute from the RIGObject class. The basic model may be extended by defining additional subclasses, as described in section 4.

Each of the classes, attributes, and relationships has a well-defined seman-

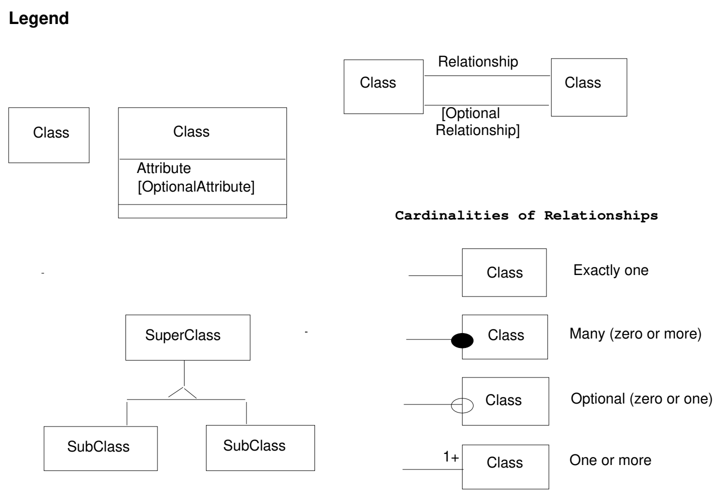

Figure 1: Legend for Data Model Notation

tics which is specified in the BIDM document. The datatype and the allowed multiplicity for each attribute are also specified.

The semantics of the UniqueID attribute for the Asset and Element classes say that it is a globally unique name used to refer to an object – e.g., for the purpose of retrieving metadata about the object or of retrieving the object itself. Global uniqueness requires that no two objects be given the same UniqueID, even object owned by different repositories. Although various proposals for the UniqueID attribute have been brought before the RIG, as yet none has been adopted. For lack of a better solution, most BIDM implementors are currently usin URLs for the UniqueID field. The RIG is monitoring progress by the Internet Engineering Task Force (IETF) on Uniform Resource Names (URNs) and may adopt URNs for the UniqueID attribute if and when URNs become a standard.

Many organizations and disciplines use controlled vocabularies for one or more of the BIDM attributes, such as Domain and Keyword. For example, several mathematical software repositories and companies use the Guide to Mathematical Software (GAMS) to classify mathematical software [2]. As described in section 4, work is underway on a model that would allow a library to indicate that is using a particular controlled vocabulary for a particular attribute.

3 Data Model Bindings

In order for catalog information to be exchanged between software repositories, the abstract data model described in section 2 is mapped to a concrete syntax

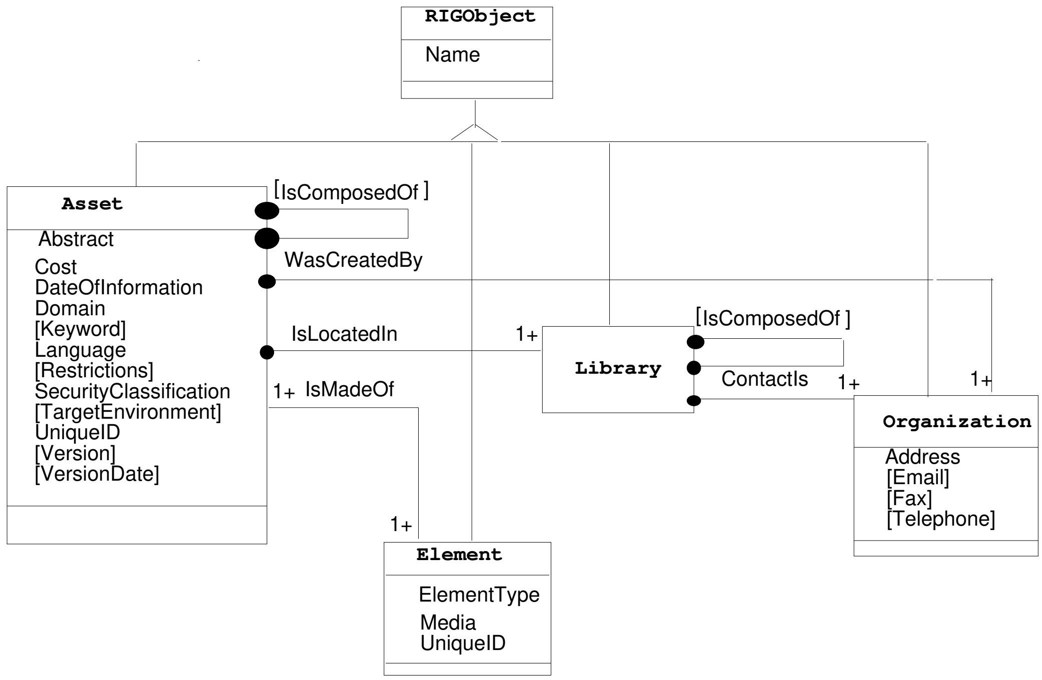

Figure 2: Basic Interoperability Data Model

that can be transferred using a file transfer protocol such as FTP or HTTP. So far the RIG has defined two such bindings, one that maps the BIDM to an SGML Document Type Definition (DTD), and another that maps the BIDM to META and LINK tags in the header of an HTML document. These bindings have been implemented and tested by RIG members. A document containing the specifications for these bindings is currently in the IEEE standardization process.

The SGML DTD for the RIG BIDM and the Asset Certification Framework extension, which is described in section 4, may be found in Appendix A. Each class, attribute, and relationship is defined as an SGML element. Subclasses are represented by nesting the subclass element within the parent class element. For any of the relationships, the implementor has the option of nesting the destination class SGML elements within the parent class SGML element, or of specifying an identifier for a separate file which contains the metadata for the destination class. An example of using the SGML binding is given in Appendix B.

With the HTML binding, the attribute and relationship metadata for an object is placed in META and LINK tags in the header of an HTML document that otherwise describes the object – for example an HTML page that describes a software asset or an organization. Examples of the HTML binding are given in Appendices C and D.

As shown by the examples, the value of a BIDM attribute may be given either by placing it in a META tag of the form

```
<META NAME="data-model.class.attribute" CONTENT="value">
```

or by placing the value in a separate file and pointing to it using a LINK tag of the form

```
<LINK REL="data-model.class.attribute"
   HREF="URL of file containing the attribute value ">
```

The latter method is useful when the value contains special characters, such as quotations marks or HTML markup, that can't occur in the CONTENT of a META tag.

The metadata for an object that is the destination of a relationship may be in-lined in the HTML file that describes the source object, with metadata for multiple destinations correlated by means of a correlation number, by using META tags of the form

```
<META NAME="data-model.src-class.rel.dest-class[.cn].attribute">
```

where `rel` stands for relationship and `cn` for correlation number. Alternatively, if another file contains the metadata for the destination of a relationship, the file may be specified by using a LINK tag of the form

```
<LINK REL="BIDM.source-class.relationship.dest-class"
      HREF="URL for destination">
```

To express an inverse relationship, the file containing the metadata for the destination of the relationship may include a LINK tag of the following form:

```
<LINK REV="BIDM.source-class.relationship.dest-class"
      HREF="URL for relationship source">
```

For example, in the metadata for the Netlib library,

```
<LINK REV="BIDM.asset.islocatedin.library"
      HREF="http://www.netlib.org/lapack/"
      TITLE="LAPACK">
```

indicates that the LAPACK software package is contained in the Netlib library.

The binding process involves some form of collection to retrieve, parse, and validate metadata located in HTML or SGML files stored on the Internet or on an organization's internal Intranet. A typical scenario would be for a library administrator to initiate a Web spider to collect and validate metadata files from outside libraries. This metadata could then be incorporated into the library's environment where it could be stored in a database or directory structure that could be searched by the library's users.

So far participants in RIG interoperability experiments have overwhelmingly chosen to use the HTML binding over the SGML binding, probably because of unfamiliarity with SGML and SGML tools. However, advantages of the SGML binding over the HTML binding include the following:

1. Metadata can be validated using an SGML parser to provide a check for correct syntax and required fields.

2. The hierarchical data model can be represented by nesting of SGML elements and thus does not need to be flattened out as in the HTML binding.

3. Existing SGML tools can be used to process the metadata files automatically.

Because the HTML and SGML bindings have been in use for less than a year, it is too earlier to tell which will end up being most widely adopted, or if a completely different binding, such as perhaps Z39.50, will prove more successful.

4 Model Extensions

Although the Basic Interoperability Data Model has greatly enhanced the ability of reuse libraries to interoperate, it is desirable to be able to extend the basic model to cover specific areas more thoroughly or to meet the needs of specialized libraries. One area for which an extension has already been defined is that of asset evaluation and certification. The extension is the RIG Asset Certification Framework, which defines a standard for the consistent structure, labeling, and description of evaluation and certification policies and results, and which is discussed further below. The RIG is working on another extension, also discussed below, called the Intellectural Property Rights Framework, which will provide a consistent framework for labeling and describing intellectual property rights and other legal restrictions on software assets. Another reason extensions are needed is that a library may have additional metadata, beyond that specified in the BIDM, that it would like to make available, and it may wish to extend the BIDM for this purpose.

Because it is expected that extensions to the basic model will be defined by groups outside the RIG, and to ensure that the RIG itself follows a consistent methodology in defining model extensions, the RIG is working on a formal meta-model for describing allowed extensions. Although the short term goal for this meta-model is thattt it be understandable and usable by human data modelers, a longer term goal is that it be understandable by intelligent agent programs that would interpret and process metadata from the basic data model and its extensions automatically.

4.1 The Asset Certification Framework

Most software reuse libraries organize their evaluation and certification policies by levels. These levels provide a quick reference for the user in determining what evaluation and certification criteria have been met by particular assets. In general, increasing levels represent increasing confidence in the asset, as well as increasing certification effort and cost. However, each library has defined its levels differently, and the different levels and policies are confusing to users of multiple interoperating libraries. Each reuse library needs to be able to define certification policies that are unique to its particular mission and that are compliant with domain-specific standards. Rather than attempting to drive all libraries to a standard set of levels, the Asset Certification Framework (ACF) prescribes a standard for organizing and describing different policies. Thus, the ACF provides a common basis for commparing different policies and for understanding different libraries' evaluation and certification activities and results.

The ACF extends the BIDM by adding the AwardedWith relationship to the Asset class of the BIDM and by defining additional classes of objects that are relevant to evaluation and certification. A pictorial view of the ACF, using the legend from section 2 and with attributes of the original BIDM classes omitted, is shown in Figure 3. A tabular view of the ACF is shown in Figure 4.

Certification quality factors are high level evaluation criteria, such as completeness, correctness, and reliability. Certification properties define features or characteristics of an asset that may be assessed as being true or false, or that may be measured. Certification methods are documented evaluation techniques,

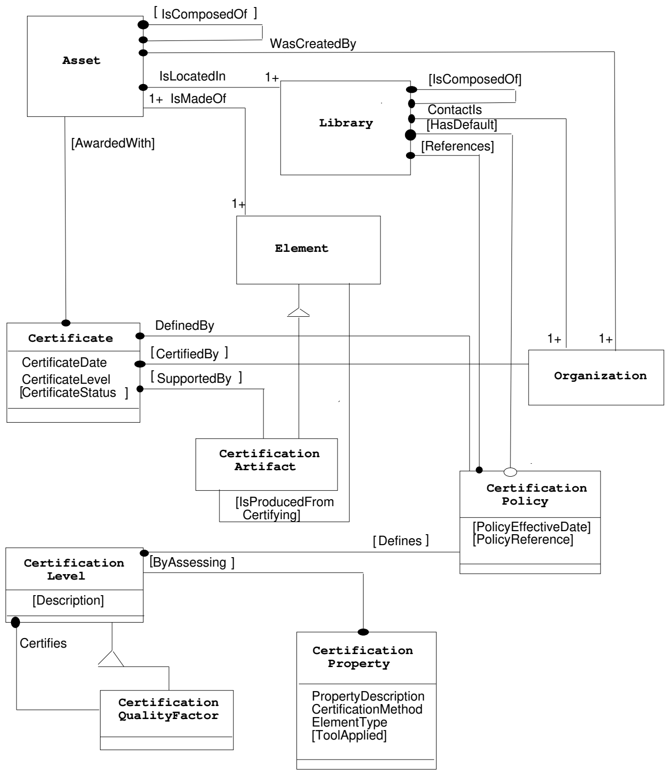

Figure 3: Asset Certification Framework

which may include compilation, static analysis, inspection, testing, formal verification, and benchmarking.

4.2 Intellectual Property Rights Framework

The RIG Technical Committee on Intellectual Property Rights is working on developing a BIDM extension for labeling assets with information regarding legal restrictions commonly asserted in the United States, such as copyright, patents, licensing, and export resrictions. The committee does not intend to deal with international issues, enforcement of legal restrictions, or advocation of changes in intellectual property rights law. The committee will consider mechanisms for asserting and enforcing legal restrictions for the purpose of describing such mechanisms, but will not develop such mechanisms. One goal of this work is to enable pre-negotiation of agreements between reuse libraries that facility large-scale sharing of restricted software – i.e., so that a separate agreement for each software asset and each pair of interoperating libraries does not need to be

Certification Levels	Certification Property		Element Type
	Property Description	Certification Method	

Figure 4: Tabular View of Asset Certification Framework

negotiated.

As an example of a legal restriction and the associated liability issues, consider export restrictions on software. Export licensing requirements for software fall under the Export Administration Regulations (EAR) and, for software related to military use, the International Traffic in Arms Regulations (ITAR). All software is export controlled – the issue is what type of export license is required. Depending on its export classification, software may be exportable under a general license, or may require a validated license which requires specific written authorization from the Bureau of Export Regulations. Export of software under a general license may or may not require a written assurance signed by the recipient that the recipient will also follow export regulations. to complicate matters further, there are lists of prohibited customers and prohibited end uses which must be honored.

The liability issues arise when one considers the penalties for not following export regulations, which can include fines and imprisonment. What type of export license applies to a piece of software and whether export license conditions have been met are legal opinions. Liabilityyy can occur when an export classification is found to be incorrect or when a license condition is found not to have been met. Now consider the case where a library imports a piece of software from another library and distributes it to its own users. If a violation of export regulations occurs, who is responsible, the originating library or the distributing library? The issue is complicated and depends on a number of factors, including who determined the export classification and what legal agreements had been made between the libraries.

The RIG hopes that by providing the means to unambiguously describe export and other legal restrictions on software, risks and fears of liability and litigation will be reduced and not unduly impede the exchange of software between libraries. Although the RIG's main concern is software, software is just one type of technical data covered by export regulations, and thus we expect our work be relevant to digital libraries in other technical domains.

4.3 Meta-Model

The approach being taken by the RIG in defining a formal model for describing model extensions is to define the allowed extensions in terms of formal data modeling notation [7]. Data modelers will thus be able to determine unambiguously how new classes, attributes, and relationships may be defined, as well as how to represent these entities in terms of the same data modeling notation.

The BIDM makes no provision for controlled vocabularies. However, it is clearly desirable for reuse libraries to be able to use existing controlled vocabularies, such as keyword lists, taxonomies, and thesauri, as well as place other constraints on values of an attribute, such as a particular date format. To meet this need, the meta-model will include a scheme for describing constraints on the possible values of an attribute.

5 The National HPCC Software Exchange

The National HPCC Software Exchange (NHSE) provides a uniform interface to a distributed set of discipline-oriented HPCC repositories [4] [1]. As such, the NHSE is a *virtual repository*, in that it catalogs and points to software maintained elsewhere, except for archive and mirror copies stored on NHSE machines. A virtual repository is a type of interoperation that involves a hierarchical relationship. The NHSE virtual repository architecture is shown in Figure 5.

In many cases, a discipline-oriented repository will wish to provide its own specialized interface to its software collection. The repository may use classification schemes and search tools tuned to its particular discipline. For example, the Netlib [5] [2] and GAMS [3] [3] mathematical software repositories use the GAMS classification scheme and are developing expert search subsystems for specific GAMS classes. Discipline-oriented repositories will also be in the best position to review and evaluate software within their own domains. In addition to providing access to its own software, a repository may wish to import software descriptions from other repositories and make this software available from its own interface. For example, a computational chemistry repository may wish to provide access to mathematical software and to parallel processing tools

[1] `http://www.netlib.org/nhse/`
[2] `http://www.netlib.org/`
[3] `http://gams.nist.gov/`

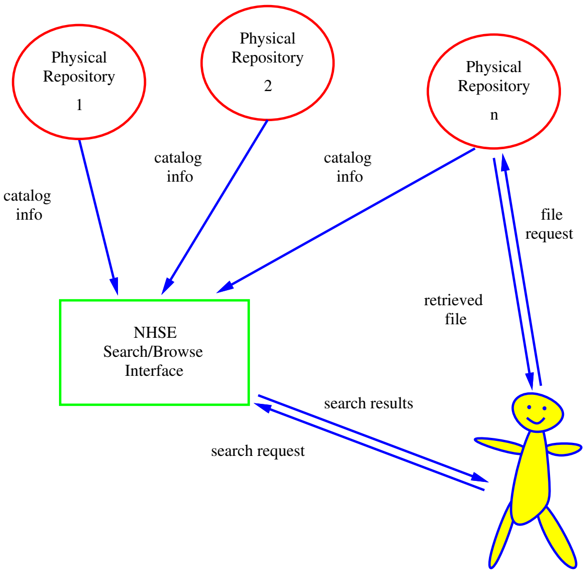

Figure 5: NHSE Virtual Repository Architecture

in a manner tuned to the computational chemistry discipline. A repository interoperation architecture is shown in Figure 6.

The NHSE is using the BIDM Web bindings described in section 3 as its interoperability mechanism. Participating HPCC repositories and some individual contributors have placed META and LINK tags in the headers of HTML files that describe their software resources. Some repositories are making use of the SGML binding as well. In addition to the BIDM fields, the NHSE data model includes a few additional fields that are desirable for NHSE interoperation. The relevant data model for a field is currently specified by prefixing the field name with the data model name in the name attribute of the META tag. In the future, NHSE extensions to the BIDM will be described using the RIG meta-model which is currently under development. The NHSE is developing a toolkit called Repository in a Box (RIB) that will assist repository maintainers 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 to their software catalog.

As a virtual repository, the NHSE sees a need for a globally unique identifier that unambiguously identifiers a particular version of a software asset. Such unambiguous identification is necessary for a number of reasons, including the following:

- version tracking

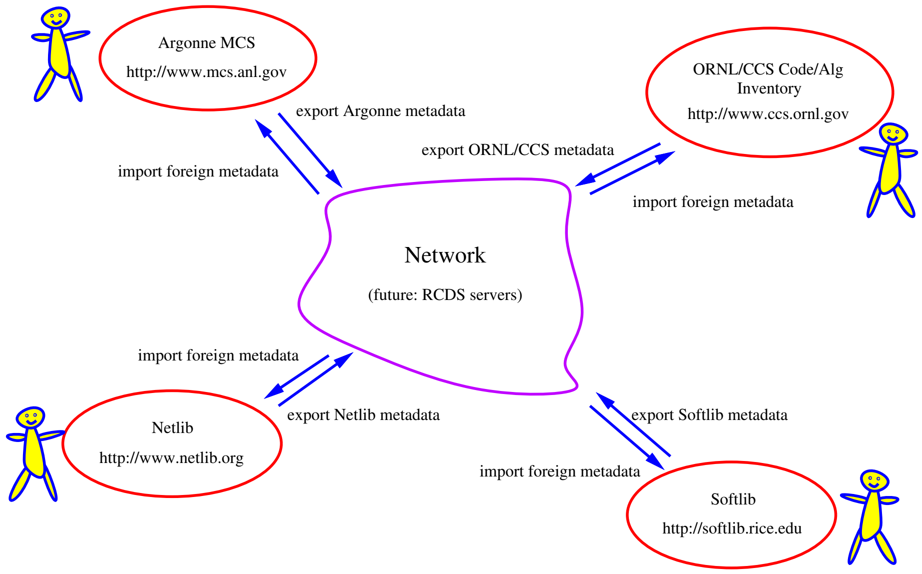

Figure 6: Repository Interoperation Architecture

- associating testing and review metadata with the exact version that was reviewed

- reporting and reproducing scientific results

However, the NHSE also sees a need for a stable name for a resource that does not change every time there is a minor bug fix or revision. The NHSE is currently experimenting with using both URLs and URNs in the metadata that is exchanged using the Web bindings of the RIG BIDM. The NHSE data model includes an additional fingerprint field for identifying the exact version of a file. The fingerprint scheme currently used by the NHSE is MD5 [6].

Distributed maintenance of resources, although desirable for maintaining information close to its source and thus allowing local control and keeping it up-to-date, raises performance and reliability problems for access by remote users. Performance and reliability problems can be solved by replication and caching. However, replication and caching raise consistency and intellectual property rights issues.

The Resource Cataloging and Distribution System (RCDS) under development at the University of Tennessee uses a consistency model based on Location Independent File Names (LIFNs). Once assigned, a LIFN is immutably bound to a particular sequence of bytes. After updating a file, a publisher assigns it a new LIFN, registers the new URN-to-LIFN binding with an RCDS catalog server, and notifies authorized file servers who can then acquire the new file and notify a location server of the new LIFN-to-URL binding. Thus, the RCDS scheme is a combination of TTL-based "pull" consistency, with file servers pulling updates at their convenience, and invalidation-based "push" updating by efficient propagation of meta-information updates among catalog servers.

The NHSE is planning to mirror authorized copies of software from the various HPCC repositories and individual software providers on NHSE file servers. The NHSE is also planning to run experimental RCDS catalog and location servers on the distributed set of NHSE servers. Experiments will be carried out to compare the performance and efficiency of the RCDS file replication approach with other proposed replication and caching schemes.

The NHSE has designed a software review policy that enables easy access by users to information about software quality, but which is flexible enough to be used across and specialized to different disciplines. The three review levels recognized by the NHSE are the following:

1. Unreviewed
2. Partially reviewed
3. Reviewed

The *Unreviewed* designation means only that the software has been accepted into the owning repository and is thus within the scope of HPCC and of the discipline of that repository. The *Partially reviewed* designation means that the software has been checked by a librarian for properties that may be verified by inspection, including completeness, adequate documentation, and good software construction. The *Reviewed* designation means that the software has been reviewed in a review article in the electronic journal *NHSE Review* [4] by an expert in the appropriate field. Domain-specific repositories and expert reviewers are expected to refine the NHSE software review policy by adding additional review criteria, evaluation properties, and evaluation methods and tools. The NHSE also provides for soliciting and publishing author claims and user comments about software quality. All software exported to the NHSE by its owning repository or by an individual contributor is to be tagged with its current review level and with a pointer to a review abstract which describes the software's current review status and includes pointers to supporting material. The review information is also encoded in terms of the RIG Asset Certification Framework described in section 4.1 for exchange with other software repositories.

Protection of intellectual property rights should not unduly impede or slow access to software. The NHSE is faced with the task of distributing and providing efficient access to HPCC software, some of which has security classifications and/or access restrictions. The NHSE is currently undertaking a study of how efficient access can be provided while meeting legal restrictions and security objectives, and without exposing third parties, such as NHSE online service

[4] `http://nhse.cs.rice.edu/NHSEreview/`

providers, to legal liability for rights infringement or violation of U.S. export law.

6 Conclusions

We hope that groups in other domains will benefit from our experiences in developing and implementing an extensible data model for the software reuse community. We believe that the extended entity-relationship data modeling technique is a powerful way of capturing and describing metadata about network-accessible resources. We also believe that the RIG has achieved the proper balance between domain-specific standardization and domain-independent standardization, by developing an abstract semantic domain-specific data model and mapping the abstract model to concrete domain-independent representations such as SGML and HTML.

In addition to be a valuable resource for the high performance computing community, the National HPCC Software Exchange provides a testbed for exploring issues related to interoperation of independent repositories, as well as quality control and intellectual property rights of network-accessible resources. The NHSE hopes to both contribute to and benefit from progress on these issues by other digital library projects.

Acknowledgments

We acknowledge the contributions of Reuse Interoperability Library Group members to the work described in this paper, in particular Ed Comer's leadership in producing the Asset Certification Framework and Robert Terry's idea for the HTML binding of the BIDM.

We acknowledge contributions of the NHSE development team, which includes researchers at Rice University, University of Tennessee, Argonne National Laboratory, Syracuse University, and California Institute of Technology.

References

[1] *IEEE Standard for Information Technology - Software Reuse - Data Model for Reuse Library Interoperability: Basic Interoperability Data Model (BIDM).* IEEE Std 1420.1, 1995.

[2] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. GAMS: A framework for the management of scientific software. *ACM Trans. Math. Softw.*, 11(4):313–355, Dec. 1985.

[3] R. F. Boisvert, J. L. Springmann, and M. L. Strawbridge. The GAMS virtual software repository. In *Proceedings of the Thirtieth Semi-Annual Meeting*, pages 68–72, Gaithersburg, MD, September 1992. Cray User Group, Five Point Editorial Services.

[4] S. Browne, J. Dongarra, S. Green, K. Moore, T. Rowan, R. Wade, G. Fox, K. Hawick, K. Kennedy, J. Pool, R. Stevens, R. Olson, and T. Disz. The National HPCC Software Exchange. *IEEE Computational Science and Engineering*, 2(2):62–69, Summer 1995.

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

[6] R. Rivest. The MD5 message-digest algorithm. *Internet Request for Comments*, 1321, Apr. 1992.

[7] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, 1991.

Appendix A. SGML DTD for BIDM and ACF

```
<!ENTITY   %  n.Identifier  "(URN | LIFN | URL | HDL)" >

<!ENTITY   %  n.Text        "(#PCDATA)" >
<!ENTITY   %  n.Date        "(#PCDATA)"
 -- YYYY-MM-DD -->
<!ENTITY   %  n.String      "(#PCDATA)" >
 -- at most 1023 characters -->
<!ENTITY   %  n.LIFN             "(#PCDATA)" >
<!ENTITY   %  n.URL              "(#PCDATA)" >
<!ENTITY   %  n.URN              "(#PCDATA)" >
<!ENTITY   %  n.HDL              "(#PCDATA)" >

<!ENTITY   %  RIGObject.attributes
              "Name" >

<!ENTITY   %  Asset.attributes
                      "Abstract, Cost,
                      DateOfInformation, DistributionStatement?,
                      Domain, Keyword*, Language+, Restrictions?,
                      SecurityClassification, TargetEnvironment*,
                      UniqueID+, Version?, VersionDate? ">

<!ENTITY   %  Asset.relationships
                      "AwardedWith*, IsComposedOf*,
                      IsLocatedIn+, IsMadeOf+, WasCreatedBy+" >

<!ENTITY   %  Element.attributes
              "ElementType, Media, UniqueID" >

<!ENTITY   %  Organization.attributes
              "Address, Email*, Fax*, Telephone*" >

<!ENTITY   %  Library.relationships
                      "ContactIs+, IsComposedOf*, HasDefault?, References*" >

<!ENTITY   %  Certificate.attributes
                      "CertificateDate, CertificateLevel, CertificateStatus?" >

<!ENTITY   %  Certificate.relationships
                      "CertifiedBy?, DefinedBy, SupportedBy*" >

<!ENTITY   %  CertificationPolicy.attributes
```

```
                    "PolicyEffectiveDate?, PolicyReference*" >

<!ENTITY   %  CertificationPolicy.relationships
                    "Defines*" >

<!ENTITY   %  CertificationLevel.attributes
                    "Description?" >

<!ENTITY   %  CertificationLevel.relationships
                    "ByAssessing*, Certifies*" >

<!ENTITY   %  CertificationProperty.attributes
                    "PropertyDescription, CertificationMethod,
                     ElementType, ToolApplied?" >

<!ENTITY   %  CertificationArtifact.relationships
                    "IsProducedFromCertifying*" >

<!ELEMENT  RIGObject O O (Asset | Element | Library | Organization |
                        Certificate | CertificationPolicy |
                        CertificationLevel | CertificationProperty) >

<!ELEMENT  Asset - - (%RIGObject.attributes;, %Asset.attributes;,
                    %Asset.relationships;) >

<!ELEMENT  UniqueID - - (%n.Identifier;) >

<!ELEMENT  Name - - (%n.String;) >

<!ELEMENT  Abstract - - (%n.Text;) >

<!ELEMENT  Cost - - (%n.Text;) >

<!ELEMENT  DateOfInformation - - (%n.Date;) >

<!ELEMENT  DistributionStatement - - (%n.Text;) >

<!ELEMENT  Domain - - (%n.String;) >

<!ELEMENT  Keyword - - (%n.String;) >

<!ATTLIST  Keyword   Scheme     CDATA    #IMPLIED >

<!ELEMENT  Language - - (%n.String;) >
```

```
<!ELEMENT  Restrictions - - (%n.Text;) >

<!ELEMENT  SecurityClassification - - (%n.Text;) >

<!ELEMENT  TargetEnvironment - - (%n.Text;) >

<!ELEMENT  Version - - (%n.String;) >

<!ELEMENT  VersionDate - - (%n.Date;) >

<!ELEMENT  IsComposedOf - - (%n.Identifier;) >

<!ELEMENT  IsLocatedIn - - (%n.Identifier; | Library) >

<!ELEMENT  IsMadeOf - - (%n.Identifier; | Element) >

<!ELEMENT  WasCreatedBy - - (%n.Identifier; | Organization) >

<!ELEMENT  LIFN  - - (%n.LIFN;)
 -- Location Independent File Name -->

<!ELEMENT  URN   - - (%n.URN;)
 -- Uniform Resource Name -- >

<!ELEMENT  URL   - - (%n.URL;)
 -- Uniform Resource Locator -- >

<!ELEMENT  HDL  - - (%n.HDL;)
 -- CNRI Handle -- >

<!ELEMENT  Element - - (CertificationArtifact |
                    (%RIGObject.attributes;, %Element.attributes;)) >

<!ELEMENT  ElementType - - (%n.String;) >

<!ELEMENT  Media - - (%n.Text;) >

<!ELEMENT  Library - - (%RIGObject.attributes;, %Library.relationships;) >

<!ELEMENT  AwardedWith - - (%n.Identifier; | Certificate) >

<!ELEMENT  ContactIs - - (%n.Identifier; | Organization) >
```

```
<!ELEMENT  Organization - - (%RIGObject.attributes;,
                              %Organization.attributes;) >

<!ELEMENT  Address - - (%n.String;) >

<!ELEMENT  Email - - (%n.String;) >

<!ELEMENT  Fax - - (%n.String;) >

<!ELEMENT  Telephone - - (%n.String;) >

<!ELEMENT  Certificate - - (%RIGObject.attributes;,
                             %Certificate.attributes;,
                             %Certificate.relationships;) >

<!ELEMENT  CertificationPolicy - - (%RIGObject.attributes;,
                                     %CertificationPolicy.attributes;,
                                     %CertificationPolicy.relationships;) >

<!ELEMENT  CertificationLevel - - (CertificationQualityFactor |
                                   (%RIGObject.attributes;,
                                    %CertificationLevel.attributes;,
                                    %CertificationLevel.relationships;)) >


<!ELEMENT  CertificationArtifact - - (%RIGObject.attributes;,
                                       %Element.attributes;,
                                       %CertificationArtifact.relationships;) >

<!ELEMENT  CertificationQualityFactor - - (%RIGObject.attributes;,
                                 %CertificationLevel.attributes;,
                                 %CertificationLevel.relationships;) >

<!ELEMENT  CertificationProperty - - (%RIGObject.attributes;,
                                 %CertificationProperty.attributes;) >

<!ELEMENT  CertificateDate - - (%n.Date;) >

<!ELEMENT  CertificateStatus - - (%n.Text;) >

<!ELEMENT  CertificateLevel - - (%n.String;) >

<!ELEMENT  DefinedBy - - (%n.Identifier; | CertificationPolicy) >
```

```
<!ELEMENT  PolicyEffectiveDate - - (%n.Date;) >

<!ELEMENT  PolicyReference - - (%n.Text;) >

<!ELEMENT  Description - - (%n.Text;) >

<!ELEMENT  ByAssessing - - (%n.Identifier; | CertificationProperty) >

<!ELEMENT  Certifies - - (%n.Identifier; | CertificationQualityFactor) >

<!ELEMENT  PropertyDescription - - (%n.Text;) >

<!ELEMENT  CertificationMethod - - (%n.Text;) >

<!ELEMENT  ToolApplied - - (%n.Text;) >

<!ELEMENT  HasDefault - - (%n.Identifier; | CertificationPolicy) >

<!ELEMENT  References - - (%n.Identifier; | CertificationPolicy) >

<!ELEMENT  SupportedBy - - (%n.Identifier; | CertificationArtifact) >

<!ELEMENT  Defines - - (%n.Identifier; | CertificationLevel) >

<!ELEMENT  CertifiedBy - - (%n.Identifier; | Organization) >
```

Appendix B. Example of Using the SGML Binding

```
<!DOCTYPE RIGOBJECT SYSTEM "BIDM.dtd" [
]>
<RIGObject>
<Asset>
  <Name>ScaLAPACK</Name>
  <Abstract>
ScaLAPACK is a library of high performance linear algebra
routines for distributed memory MIMD computers.  It is a
continuation of the LAPACK project, which designed and
produced analogous software for workstations, vector
supercomputers, and shared memory parallel computers.  Both
libraries contain routines for solving systems of linear
equations, least squares problems, and eigenvalue problems.
Most machine dependencies are limited to two standard
libraries called the BLAS, or Parallel Basic Linear Algebra Subroutines,
and the BLACS, or Basic Linear Algebra Communication Subroutines.
LAPACK and ScaLAPACK will run on any machine where the PBLAS
and the BLACS are available.
  </Abstract>
  <Cost>free</Cost>
  <DateOfInformation>1996-04-30</DateOfInformation>
  <Domain>numerical/linear algebra</Domain>
  <Language>Fortran 77</Language>
  <SecurityClassification>None</SecurityClassification>
  <TargetEnvironment>
PVM 3.3 or later; Intel distributed
memory computers; IBM SP series; Thinking Machines CM-5.
  </TargetEnvironment>
  <UniqueID><URN>urn:inet:netlib.org:scalapack</URN></UniqueID>
  <UniqueID><URL>http://www.netlib.org/scalapack</URL></UniqueID>
  <AwardedWith>
    <Certificate>
      <Name>Review abstract for Scalapack</Name>
      <CertificateDate>1995-08-21</CertificateDate>
      <CertificateLevel>Partially reviewed</CertificateLevel>
      <CertifiedBy>
        <Organization>
          <Name>National HPCC Software Exchange</Name>
          <Address>University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996
```

```
        </Address>
        <Email>nhse-librarian@netlib.org</Email>
      </Organization>
    </CertifiedBy>
    <DefinedBy>
      <URL>http://www.netlib.org/nhse/software_submit/review.sgml</URL>
    </DefinedBy>
    <SupportedBy>
      <CertificationArtifact>
        <Name>Partial Review Report for Scalapack</Name>
        <ElementType>Text file</ElementType>
        <Media>Electronic</Media>
        <UniqueID><URL>
          http://www.cs.utk.edu/~rowan/nhse-partial-reviews/scalapack
        </URL></UniqueID>
      </CertificationArtifact>
    </SupportedBy>
  </Certificate>
</AwardedWith>
<IsLocatedIn>
  <Library>
    <Name>Netlib</Name>
    <ContactIs>
      <Organization>
        <Name>Netlib</Name>
        <Address>University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996
        </Address>
        <Email>netlib_maintainers@netlib.org</Email>
      </Organization>
    </ContactIs>
  </Library>
</IsLocatedIn>
<IsMadeOf>
  <Element>
    <Name>ScaLAPACK tar file containing source code and installation
    guide for the ScaLAPACK routines and the PBLAS routines.
    Assumes the BLAS and the BLACS are available.
    </Name>
    <ElementType>Source code</ElementType>
    <Media>Electronic</Media>
    <UniqueID>
      <URL>http://www.netlib.org/scalapack/scalapack.tar.z</URL>
    </UniqueID
  </Element>
```

```
  </IsMadeOf>
  <WasCreatedBy>
    <Organization>
      <Name>ScaLAPACK Development Group</Name>
      <Address>University of Tennessee,107 Ayres Hall, Knoxville, TN 37996
      </Address>
      <Email>scalapack@cs.utk.edu</Email>
    </Organization>
  </WasCreatedBy>
</Asset>
</RIGObject>
```

Appendix C. Example of Using the HTML Binding for an Asset

In http://www.netlib.org/scalapack/:

```
<head>
<TITLE>ScaLAPACK</TITLE>
<meta name="BIDM.asset.name" content="ScaLAPACK">
<meta name="BIDM.asset.abstract" content="
ScaLAPACK is a library of high performance linear algebra
routines for distributed memory MIMD computers.  It is a
continuation of the LAPACK project, which designed and
produced analogous software for workstations, vector
supercomputers, and shared memory parallel computers.  Both
libraries contain routines for solving systems of linear
equations, least squares problems, and eigenvalue problems.
Most machine dependencies are limited to two standard
libraries called the BLAS, or Parallel Basic Linear Algebra Subroutines,
and the BLACS, or Basic Linear Algebra Communication Subroutines.
LAPACK and ScaLAPACK will run on any machine where the PBLAS
and the BLACS are available.">
<meta name="BIDM.asset.cost" content="free">
<meta name="BIDM.asset.targetenvironment" content="
PVM 3.3 or later; Intel distributed
memory computers; IBM SP series; Thinking Machines CM-5."
<meta name="BIDM.asset.domain" content="numerical/linear algebra">
<meta name="NHSE.asset.contactis.organization.email" content="scalapack@cs.utk.e
du">
<meta name="BIDM.asset.keyword" content="D. Linear algebra">
<meta name="BIDM.asset.keyword" content="parallel numerical library;
distributed memory multiprocessor; MIMD machine">
<link rel="NHSE.asset.reference"
  href="http://www.netlib.org/lapack/lawns/lawn100.ps">
<link rel="BIDM.asset.UniqueID" href="http://www.netlib.org/scalapack/"
urn="urn:inet:netlib.org:scalapack">
<link rel="BIDM.asset.islocatedin.library"
title="Netlib" href="http://www.netlib.org/"
urn="urn:inet:netlib.org:netlib">
<link rel="BIDM.asset.awardedwith.certificate"
href="http://www.netlib.org/nhse/sw_catalog/num/linalg/scalapack_review.html">
</head>
```

Appendix D. Example of Using the HTML Binding for ACF Metadata

In http://www.netlib.org/nhse/sw_catalog/num/linalg/scalapack_review.html:

```
<head>
<title>NHSE Review Abstract for Scalapack</title>
<meta name="BIDM.Certificate.CertificateLevel" content="Partially reviewed">
<meta name="BIDM.Certificate.SupportedBy.CertificationArtifact.Name"
content="Scalapack Partial Review Report"
<link rel="BIDM.Certificate.SupportedBy.CertificationArtifact.UniqueID"
href="http://www.cs.utk.edu/~rowan/nhse_partial_reviews/scalapack">
<link rel="BIDM.Certificate.DefinedBy.CertificationPolicy"
title="NHSE Software Review Policy"
href="http://www.netlib.org/nhse/software_submit/review.html">
<link rev="BIDM.asset.awardedwith.certificate"
href="http://www.netlib.org/scalapack/">
</head>
```