Technologies for Repository Interoperation and Access Control[*]

Shirley Browne, Jack Dongarra, Jeff Horner, Paul McMahan, and Scott Wells
Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301

ABSTRACT
Over the past several years, network-accessible repositories have been developed by various academic, government, and industrial organizations to provide access to software and related resources. Allowing distributed maintenance of these repositories while enabling users to access resources from multiple repositories via a single interface has brought about the need for interoperation. Concerns about intellectual property rights and export regulations have brought about the need for access control. This paper describes technologies for interoperation and access control that have been developed as part of the National High-performance Software Exchange (NHSE) project, as well as their deployment in a freely available repository maintainer's toolkit called Repository in a Box. The approach to interoperation has been to participate in the development of and to implement an IEEE standard data model for software catalog records. The approach to access control has been to extend the data model in the area of intellectual property rights and to implement access control mechanisms of varying strengths, ranging from email address verification to X.509 certificates, that enforce software distribution policies specified via the data model. Although they have been developed within the context of software repositories, these technologies should be applicable to distributed digital libraries in general.

KEYWORDS: Interoperation, intellectual property rights, access control, authentication, software licensing, data modeling, software reuse, standards

INTRODUCTION

Our research on repository interoperation and access control technologies has been conducted within the context of the National High-performance Software Exchange (NHSE) project. The NHSE provides a uniform interface to a distributed set of discipline-oriented High Performance Computing and Communications (HPCC) repositories [1] [4]. Development of the NHSE has been funded by the federal HPCC agencies with the goal of sharing and distributing software and related resources developed by the federal HPCC program among government and academic researchers and industry. The NHSE is a *virtual repository* in that it catalogs and points to software and related resources (e.g., documentation, test suites, technical reports) 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. Interoperation also takes place between distributed discipline-oriented repositories, either within the same discipline or between disciplines. Some repositories are organized along administrative boundaries, rather than disciplinary boundaries, and the user benefits from a virtual discipline-oriented view created on top of the underlying organization repositories. In all of these cases, it is desirable that repositories be able to interoperate easily.

Three disciplines the NHSE has targeted initially are parallel systems software and tools, high performance math software, and computational chemistry. The parallel systems software and tools repository, called PTLIB, provides comprehensive, up-to-date coverage on compilers, communication libraries, debuggers, and performance analyzers for distributed and shared memory parallel computers. HPC-netlib provides access to high performance math software. A computational chemistry repository has been developed as an example of a repository for an application area. Additional discipline-oriented repositories are under development by the Department of Defense High Performance Computing Modernization Program, the NASA Earth and Space Sciences program, and the NSF-sponsored metacomputing centers (NCSA and NPACI).

The advantages of interoperation are many. Interoperation permits distributed administration of specialized repositories, while allowing users to access resources from multiple repositories via a single interface. Repositories can reduce the

[*]This work is supported by the National Aeronautics and Space Administration under Grant NAG 5-2736 and by the Defense Advanced Research Projects Agency under grant number DAAH04-95-1-0595, administered by the U.S. Army Research Office

[1]`http://www.nhse.org/`

burden of storing redundant contents by simply pointing to resources stored in other repositories. Specialization means that resources are maintained by discipline experts who are the best qualified to select, evaluate, catalog, and provide access to resources in their disciplines.

The NHSE has the goal of distributing HPCC software to as broad a U.S. audience as possible, so as to maximize the return on HPCC agency investment in development of this software by promoting further research and technology transfer and increasing U.S. market competitiveness. Where possible, HPCC software should also be made available to foreign researchers who are collaborating with U.S. scientists. However, the distribution mechanisms must provide reasonable assurances that intellectual property rights are protected and that export regulations are abided by. The role of the NHSE has been to provide technological solutions that enable compliance with U.S. laws, that satisfy the HPCC agencies' rights management and software distribution policies, and that protect both owners and distributors of access controlled resources from liability and prosecution.

The following sections describe technologies developed by the NHSE for interoperation and access control, as well as the implementation and deployment of these technologies in the Repository in a Box toolkit that is freely available to organizations and individuals who wish to set up their own interoperable, secure repositories. Interoperation and access control are related in that mechanisms are needed that facilitate the exchange, understanding, and negotiation of different rights management policies between different interoperating repositories. Although these technologies have been developed in the context of distributed software repositories, we believe they are applicable to the distributed operation of digital libraries in general.

APPROACH TO INTEROPERATION

Resources can be shared between interoperating repositories at two levels: 1) at the level of catalog information that describes the resources, 2) at the level of the actual resources. Advantages of the second level may be provision of faster and more reliable service to users, as well as a single point of contact for administrative procedures such as license agreements. Problems with the second level include liability for enforcing legal restrictions and proper crediting of download and usage statistics to the originating site.

The NHSE approach to sharing catalog information has been to adopt the Basic Interoperability Data Model (BIDM) developed by the Reuse Library Interoperability Group (RIG), of which the NHSE has been an active member. The BIDM is an IEEE standard that specifies a minimal set of catalog information that a software repository should provide about its software resources in order to interoperate with other repositories [2]. 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 those libraries. Repositories 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.

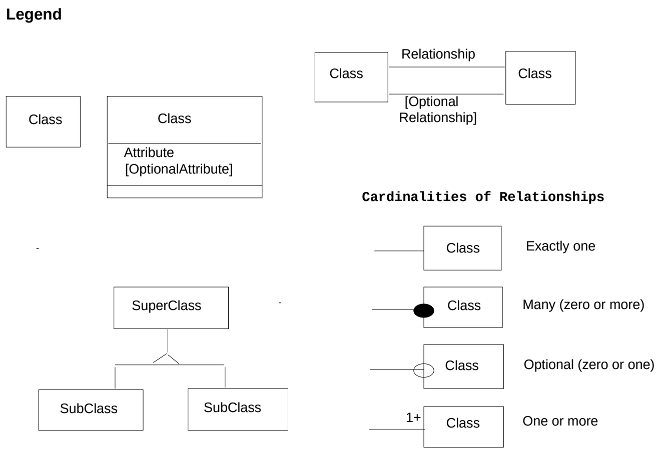

Figure 1: Legend for Data Model Notation

The BIDM may be visualized using the graphic notation of James Rumbaugh's *Object-Oriented Modeling and Design* [9]. 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. Each of the classes, attributes, and relationships has a well-defined semantics which is specified in the BIDM document. The datatype and allowed multiplicity for each attribute are also specified.

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.

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

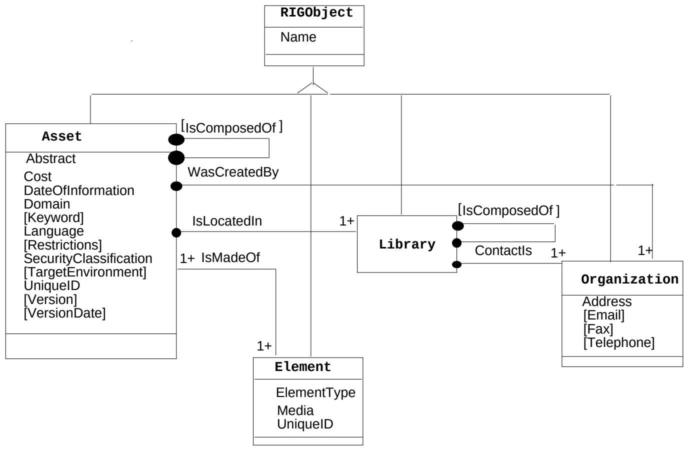

Figure 2: Basic Interoperability Data Model

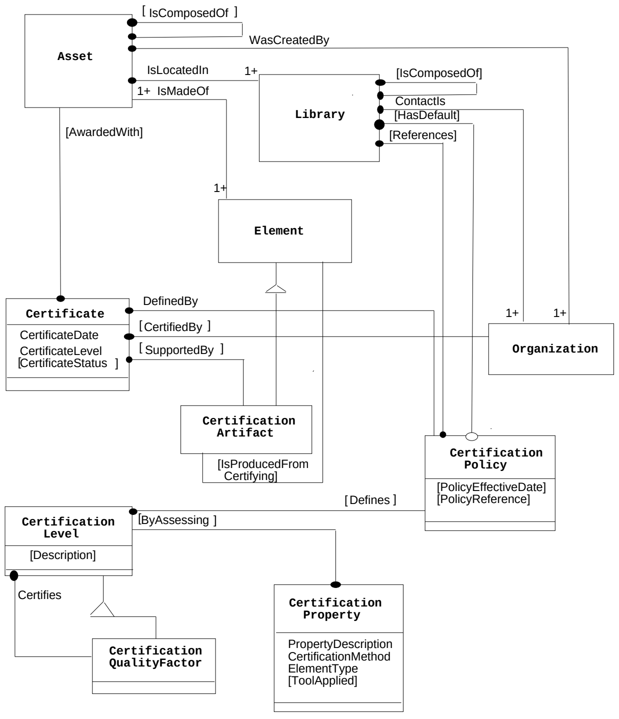

Figure 3: Asset Certification Framework

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 comparing 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, with attributes of the original BIDM classes omitted, is shown in Figure 3. 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, which may include compilation, static analysis, inspection, testing, formal verification, and benchmarking.

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: *Unreviewed*, *Checked*, and *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 *Checked* designation means that the software has been checked by a librarian for conformance with the NHSE software guidelines. The *Reviewed* designation means that the software has been reviewed 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 for their specific domain. 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 encoded in terms of the RIG Asset Certification Framework for exchange with other software repositories.

A RIG technical committee led by the NHSE has defined another extension, called the Intellectual Property Rights Framework, which provides a consistent framework for labeling and describing intellectual property rights and other legal restrictions on software assets. Similar to the Asset Certification Framework, the IPRF provides a common framework for interoperating libraries to describe and exchange their rights management policies. Similar to how asset certificates are linked to the certification policy that defined them and to the organization that did the certification in the ACF, rights assessments and licensing terms are linked to their defining policies and responsible organizations in the IPRF.

The proposed RIG IPRF has been completed and is ready for submission to the IEEE balloting process. One goal of this work is to enable pre-negotiation of agreements between reuse libraries that facilitate large-scale sharing of restricted

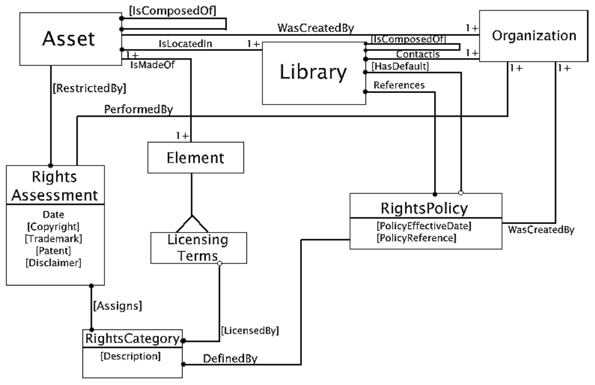

Figure 4: Intellectual Property Rights Framework

software – i.e., so that a separate agreement for each software asset and each pair of interoperating libraries does not need to be negotiated. 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. A pictorial view of the proposed IPRF is shown in Figure 4.

In order for catalog information to be exchanged between software repositories, the abstract data model described above is mapped to a concrete syntax that can be transferred using a file transfer protocol such as FTP or HTTP. Thus 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.

In the SGML DTD for the RIG BIDM and the Asset Certification Framework, 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.

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. The value of a BIDM attribute is placed in a META tag of the form

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

The metadata for an object that is the destination of a relationship is contained in another file which is specified using a LINK tag of the form

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

REPOSITORY IN A BOX

The NHSE has developed the Repository in a Box (RIB) toolkit that enables repository developers to create and maintain software catalog records using the BIDM, to exchange those records as well as software resources themselves with other repositories, and to provide a user interface for their software catalog. Multiple repositories may be maintained at the same site using a single RIB installation.

RIB provides a Web-based administrative interface for the following tasks:

- creating and maintaining software catalog records
- exchanging catalog records with other repositories, including export to the NHSE virtual repository
- setting up browsing and searching interfaces to the repository
- discipline-oriented and site-specific customization
- file uploading and mirroring

Although access control is not configurable from the Web interface, RIB also provides a mechanism for configuring separate read and write access control on a per-repository basis.

RIB allows a repository maintainer to customize the BIDM by changing the RIB's BIDM configuration file. Three example configuration files are included with RIB:

1. a simple version that contains only the Asset and Organization classes

2. the full BIDM version

3. the NHSE extension of the BIDM

By default, RIB uses the simplified version of the BIDM. The repository maintainer may alternatively select one of the other examples, or may devise his or her own version of the BIDM. Some restrictions are imposed on what modifications may be made to the BIDM configuration so that all NHSE repositories can interoperate by exchanging BIDM catalog records. You may change whether a field is required or optional. You may also change the allowed multiplicity of a field. If you do not wish to use a BIDM field, it may be omitted. You may also omit a BIDM or ACF class that you do not wish to use, although in that case you must be careful never to specify any relationships to that class in a catalog record. You may add attributes and relationships to the existing BIDM classes, and you may create new classes. A repository should not use an existing BIDM field for a different meaning than it is intended. Furthermore, if a field already exists for a given purpose, then a repository should not create another field for that purpose with a different name. Also, some BIDM fields, such as the **domain** field, are used specially by RIB and should not be omitted. The configuration file allows the maintainer to specify an alternative name for a field that will be the name actually visible to the user, and to hide a field – i.e., to have the values for a field present in the repository database but not visible to users.

The **domain** field is another place where customization may be done. With RIB, the domain hierarchy and the domain values in the software catalog records determine how the software catalog table of contents is organized. Although a default domain hierarchy is included with RIB for the general HPCC domain, RIB allows a repository creator to specify a discipline-specific domain hierarchy that provides the controlled vocabulary to be used for the domain field.

The RIB interface provides easy-to-use forms for filling in catalog information for each of the BIDM classes. Alternatively, the catalog information for an object (e.g., asset or other class) can be imported from another repository by specifying the URL that contains the HTML binding of the catalog information for the object. Once catalog information has been created and stored in the database for a repository, a searchable, browsable HTML catalog can be created, using the domain hierarchy described above to organize the table of contents. The table of contents contains one-line descriptions of the assets and points to HTML versions of the catalog information for the individual assets. A searchable index and a search form are also created automatically.

The RIB Web interface allows the maintainer to enter a URL for a file to be uploaded to the repository. The maintainer may optionally specify that a file be a mirror copy and may periodically request that mirror copies be checked for updates. Because these operations can take a long time, file uploading and checking of mirrored copies are done by a background process, with email notification sent to the maintainer when the operation completes. After the maintainer uploads a file, RIB provides the option of creating an Element object to catalog that file, with pertinent portions of the Element form already filled in.

No access controls are enforced by a RIB installation by default. Instead, access control for RIB is currently activated by enabling the access control mechanisms of the HTTP server. The actual mechanisms for access control vary depending on what HTTP server is being used, but as long as the HTTP server allows access control based on directory name it should be possible to configure the RIB installation to meet fairly rigid security requirements. If the HTTP server supports what is called Basic HTTP Authentication, then a minimum level of security can be achieved. If the HTTP server supports features such as SSL (Secure Sockets Layer) or X.509 certificates, then much higher levels of security can be achieved. More advanced access control and authentication mechanisms that are currently under development for RIB are discussed in the section entitled ACCESS CONTROL PROTOTYPE.

RIB is currently deployed and in use at three of the Department of Defense Major Shared Resource Centers (MSRCs) for setting up repositories for several of the DoD Computational Technology Areas (CTAs). Some of the CTAs, for example Computational Chemistry and Materials (CCM) are present at more than one MSRC and thus a distributed repository with a RIB at each site is being set up. However, interoperation will allow one virtual CCM repository interface to be presented to CCM users. Other sites that are using RIB to set up software repositories include the NASA Earth and Space Sciences program and the NSF-sponsored metacomputing centers at NCSA and at San Diego.

ACCESS CONTROL TECHNOLOGIES

A rights assessment of an asset is carried out by the organization that owns the asset to determine how the asset fits into its software distribution scheme. Rarely is the distribution scheme designed specifically for each individual asset. A general scheme of how each category in a classification is distributed simplifies the addition of new assets by stipulating for each class what security measures should be taken and what terms should be included in the license. Software to be distributed from NHSE-affiliated repositories generally falls into one of the following categories, although variations exist according to the software distribution policies of individual agencies and organizations:

- public domain software and documentation
- software requiring limited licensing restrictions, but still freely distributable within those restrictions
- commercial software or other software requiring fees, royalties, or other types of payments (This software typically requires licenses stipulating strong restrictions on copying, use, and redistribution of the software).
- Software technologies or documentation that fall under the control of Federal Export Regulations

Once a classification scheme has been determined, distribution tools can manage categories according to their authorization, authentication, and licensing requirements. Authorization means making a judgment as to whether or not a user should be permitted access. Authentication means verifying that the party attempting access is who he or she claims to be. The initial authorization and authentication process may require manual intervention to check the user's credentials and determine access privileges. Pre-authorization for one or more assets or entire classes of assets may be carried out once and for all, with subsequent access requiring only authentication which may be done automatically. Most electronic methods of authorization and authentication have not been tested in court. However, new legislation recognizing encryption methods such as digital signatures provides substantial legal backing.

Mass market software instigated the use of shrinkwrap to replace the necessity of having an end-user sign a license. This practice has been challenged in several court cases. However, in the most recent decision in *Pro CD v. Zeidenberg*, June 1996, the Seventh Circuit Court ruled that shrinkwraps are enforceable contracts, overturning a previous ruling. Barring a ruling from the Supreme Court, the Seventh Circuit Court ruling validates the use of shrinkwrap and electronic shrinkwrap. The validity of electronic shrinkwrap has not been explicitly questioned in court. However, electronic shrinkwrap, known by the new buzzword "clickwrap", provides extra comfort for the courts by creating an interaction with the user. The action of the user accepting the terms of the license by clicking a button and the record of that action provide an indication of commitment on the part of the user. A record of the transaction can be strong evidence against an end user's misuse of downloaded software [7].

To further computerize contracts, thirty-nine states have pending or passed legislation validating electronic or digital signatures. The language of some legislation confuses electronic and digital signatures by not making the distinction that electronic signature is simply keystrokes with the intent to simulate a signature while digital signatures use encryption keys. Nonetheless this legislative activity increases the legal binding of on-line contracts and helps support their use for software licenses.

A number of options are available for restricting access to software. A simple method would be to test the hostname of the machine from which a request originates. The hostname could be used to determine whether or not the request was made from a host within a certain domain, such as `.gov` or `.edu`. Access to software could then be either allowed or denied based on the domain name. This type of access control is simple to implement but presents some problems. One problem is that the partitioning of hosts created by domain names would not necessarily match the restriction criteria. For example, access permission to software is often based on the country from which the request originates. However, some domain names, such as `.org`, `.com`, and `.net`, do not indicate where the host is geographically located. Even if one chose to err on the side of caution and allow access only by those hosts that are partitioned correctly by domain names, the whereabouts of the person who initiated the download would still be in question. A host with access privileges might be used merely as a waystation for software headed towards a host in another domain. There are also many "hacker tools" that could be used to break into or impersonate hosts from trusted domains. Another problem is that a browser can be set to use a proxy server to fetch documents, and the server doing the access restriction will only know about the domain name of the proxy, not the real user. Despite its weaknesses, access control by domain name is currently in use at MIT for distributing PGP, at Lucent Technologies for distributing Inferno, and at NCSA for distributing cryptographically-enhanced versions of NCSA httpd and Mosaic.

Username/password access restriction is another possible option for restricting access. With this method, a user is asked to enter a correct username and password before being allowed to download a file. In the case of an HTTP server, this type of access restriction is implemented by means of a configuration file which may be either global or directory-specific. In the case of an FTP server, accounts are set up for the allowed users on the file server machine, and access permission bits and ownership of files are set appropriately. With both HTTP Basic Authentication and commonly used FTP applications, passwords are sent over the network unencrypted. In HTTP MD5 Message Digest Authentication, the password is not sent over the network at all. Rather, a "digest" that is generated based on the password and other information about the request is hashed using MD5 and sent over the network. Digest Authentication is more secure over the network, but requires more rigorous security on the server machine, because the stored information cannot be encrypted with a one-way function, whereas with Basic Authentication the server stores the password using a one-way encryption function.

Public key cryptography provides the most secure form of access control. With this method, both the request for the software and the software itself are encrypted so that they cannot be read by anyone but the intended recipient. This method is intended to be combined with a public key authentication mechanism such as PGP [6], X.509 [1], or SDSI[8]. The request would take the form of a license agreement that the user signs using his or her public key to indicate agreement to the terms and conditions for using the software. The user's public key also identifies him or her so that the software server can check whether or not that user is authorized to obtain the software. The software itself would be best encrypted using a symmetric session key which would be generated for the purpose of this transmission only.

An X.509 certificate binds an identity to a pair of electronic keys that can be used for encrypting and signing digital information. The pair consists of two related keys – a public key and a private key. The public key can be used by anyone to verify a message signed with the private key or to encrypt a message that can only be decrypted using the private key. The private key must be kept secure and protected against unauthorized use.

Certificates are issued by a Certification Authority (CA), which is a trusted party that vouches for the identity of those to whom it issues certificates. In order to prevent forged certificates, the CA's public key must be trustworthy. The CA can either widely publicize its public key or provide a certificate from a higher level CA which attests to the validity of its public key. The latter leads to a hierarchy of CAs. To obtain a certificate, an individual generates his own key pair and sends the public key to the CA with proof of his identity. Different CAs may issue certificates with different levels of identification requirements. For example, Verisign is a commercial CA that offers four classes of certificates, with increasing levels of assurance (and cost) [2]. Using the requirements for the particular level applied for, the CA checks the identification and then sends the requester a certificate attesting to the binding between the requester and his public key, along with (possibly) a hierarchy of certificates verifying the CA's public key.

An individual can use a certificate to identify himself to secure servers such as membership-based or access-controlled Web servers. Multiple certificates can be attached to a message or transaction, forming a certificate chain in which each certificate attests to the authenticity of the previous certificate. The top-level CA in the chain must be independently known and trusted by the recipient. When installed in a Web browser, a certificate functions as electronic credentials, eliminating the need for typing in a username and password. Similarly, a secure Web server uses its own certificate to assure clients that the server is run by the organization claimed and to verify the integrity of the provided documents.

X.509 client certificates are currently supported by Netscape in its Navigator 3.0 browser. Microsoft has also announced support for X.509 certificates in its client applications. X.509 server certificates are currently used in server products from IBM, Microsoft, Netscape, OpenMarket, and Oracle.

ACCESS CONTROL PROTOTYPE

As part of our research on access control, we have implemented a prototype of an access control tool that demonstrates authentication of users, electronic licensing, and integrity checking of downloaded files. The access control prototype has been implemented in collaboration with colleagues at Rice University and at Vannevar Corporation in Houston, Texas. The prototype is currently being demonstrated to government agencies to get their feedback. The prototype system consists of the following components:

- **HTML GUI**. The GUI consists of a collection of screens that guide the user though the software selection and downloading process. The GUI is implemented in a template-based scripting system and contains 'hooks' so that organizations can customize it for their own use.
- **Database interface**. The database interface software enables a repository's Web server to communicate with an underlying database that stores data about software and users and handles queries. The database collects and maintains information on users of the software and provides an administrative interface to view this information. The database also stores MD5 checksums of the repository's files, and users may access this database of checksums to verify the integrity of downloaded files.

Other components that are not part of the prototype but need to be interfaced with it are the following:

- **HTTP server**. The system is designed to be compatible with any common HTTP server.
- **DBMS**. The system uses the ODBC open database standard, so that the Database Interface can connect to most common DBMS's without modification.
- **Certificate server**. An X.509 certificate server such as the Netscape Certificate Server allows organizations to issue, sign, and manage standard X.509v3 certificates for their public key infrastructure. A certificate server could be run by a repository or by a third party that is trusted by the repository.

The current prototype demonstrates different levels of access control and authentication required for two classes of software – 1) freeware that has a license but requires only a minimal level of authentication, 2) export-restricted software that requires public-key authentication of the recipient. In both cases, the user browses or searches the software catalog to select the desired piece of software. In the first case, the user is asked to register his or her name and email address and is then presented with a license agreement. As a result of registering, the user is sent a username and password to his or her email address. After clicking a button to agree to the license conditions, the user is allowed to enter the username and password, and he or she is then taken to a download screen that allows downloading of the selected software. For this class of software, the prototype only logs the transaction and verifies that the user has entered a valid email address.

For the second class of software, the prototype requires that the user has been issued an X.509 certificate which verifies his or her identity before being allowed to download the software. A form is provided that allows the user to apply for a certificate. The Netscape Navigator browser supports an HTML tag, <KEYGEN>, which is embedded in the HTML form. When submitting a form using this browser, the browser generates an RSA key pair whose public key is digitally signed and sent to a CGI script on the Certification Authority's machine. The CGI script which runs on the Certification Authority's machine processes the input from the HTML form and places the request for a certificate into a queue. The queue is routinely checked and each request is then accepted or denied based on some set of policies set forth by the entity which controls the release of the requested software. If the request was accepted, then the Certification Authority creates and digitally signs a certificate which can be used to download the requested software. The Certification Authority contacts the party who requested the software and points them to a URL to retrieve the new certificate. The new certificate contains the public key that was generated by the <KEYGEN> tag. The party who requested the software points their browser at the URL that was provided by the Certification Authority. The browser uses the public key encoded in the certificate to associate the certificate with the appropriate private key in its local key database. The certificate has now been installed in the browser. When an HTTP server wants to require authentication based on client certificates, it uses the Secure Sockets Layer (SSL) protocol to negotiate the transfer between the browser and the HTTP

[2] `http://www.verisign.com/`

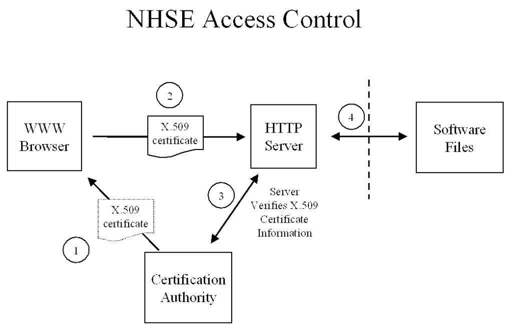

Figure 5: Access Control using X.509 Certificates

server. The HTTP server is set up so that it will only accept requests that are signed by a certain Certification Authority. When the server asks for a certificate, the browser user is prompted to choose which certificate that he or she wants to send. Depending on whether or not that certificate has been signed by the proper Certification Authority, access to the software is either accepted or denied. For purposes of the prototype, Vannevar Corporation has been acting as the Certification Authority.

The access control prototype is illustrated in Figure 5.

CONCLUSIONS AND FUTURE WORK

Through its research and development of interoperation and access control technologies, the NHSE project has provided mechanisms for sharing software related resources across organizational and disciplinary boundaries, while ensuring that access restrictions dictated by organizational software distribution policies and by government regulations are enforced. The interoperation technologies we have developed have been successfully deployed in the Repository in a Box (RIB) toolkit which is now in use at a number of high-performance computing sites. The access control technologies have been implemented and tested in a prototype. Our next step will be to integrate the access control technologies into RIB and field test the combined system. The integration will involve the following:

1. developing encodings of software distribution policies, distribution categories, and electronic licenses in terms of the IPRF extension to the BIDM,

2. developing forms for inputting IPRF catalog information into the RIB database, and

3. automatically generating the necessary on-line software accessing and licensing forms from the IPRF catalog information.

Furthermore, decisions will need to be made by organizations that use RIB about the level of access control required. If the X.509 certificate approach is chosen by an organization, then either that organization or a trusted third party will need to serve as a Certification Authority.

As pointed out in [3], the access management problem involves complex issues of contracts and interactions between the various parties. Furthermore, user roles, asset attributes, and rights policies can change over time, necessitating the dynamic association of a policy with a user and a specific asset at the time the user wishes to access the asset. Similar to the approach in [3], the IPRF keeps rights policies separate from assets, so that changes of policy can be implemented without necessarily altering the rights assessment metadata for each asset. Although the IPRF does not explicitly model user roles, such roles can be specified in the licensing terms for an asset or a rights category and can be encoded in the certificate issued to a user by a certification authority, which can be re-issued if necessary to assign new roles. As future work, we plan to work with federal agencies and laboratories on implementing their software rights management policies in terms of the IPRF and the access control mechanisms described above. We believe that our approach is sufficiently expressive and flexible to be able to interoperate with the approach described in [3], and we think that such interoperation would make an interesting experiment.

It is our opinion that the approach of developing a domain-specific abstract semantic data model, and then binding that model to an existing concrete syntax, such as SGML or HTML, provides the right level of standardization for semantic interoperability between repositories. After our presentation on it at the World Wide Web Consortium sponsored Distributed Indexing/Searching Workshop (DISW) held in Boston in May 1996, this approach was also adopted for use with the Dublin Core for cataloging Internet-accessible documents [5]. Another future goal is to formalize the method for extending the BIDM so that digital libraries outside the realm of software repositories can take advantage of the technologies we have developed, and of RIB if desired, to implement their own interoperable, access-controlled repositories. Starting from the root class RIGObject of the BIDM, classes for other objects besides software could easily be defined. The Reuse Library Interoperability Group (RIG) that authored the BIDM has evolved into the Reuse Steering Committee (RSC) [3] under the IEEE Software Engineering Standards Committee. The NHSE is working with the RSC to develop a formal sub-classing mechanism that will allow extensions to the BIDM to be clearly defined.

In addition to software repositories, distributed repositories have been set up for other types of resources such as data archives, technical reports, images, and audio and video clips. For example, the Networked Computer Science Technical

[3] `http://rsc.asset.com/`

Reports Library (NCSTRL) project [4] provides access to a distributed collection of computer science technical reports. It is desirable to be able to search across different types of repositories to retrieve all the resources relevant to one's information need, regardless of their type. One way to achieve this goal would be to define an open architecture for distributed digital library interoperation that enabled sharing at the semantic level of catalog information and resources. We are interested in contributing to the development of such an architecture and in involving the software repositories we have helped deploy in a distributed testbed for implementing and testing interoperation mechanisms between different kinds of repositories.

REFERENCES

1. *CCITT. Recommendation X.509: The Directory - Authentication Framework.* 1988.

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

3. W. Y. Arms. Implementing policies for access management. *D-Lib Magazine*, Feb. 1998. http://www.dlib.org/.

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

5. S. Browne, K. Hohn, and T. Niesen. Extensible domain-specific metadata standards. In *WWW Consortium Distributed Indexing/Searching Workshop*, Boston, Massachusetts, May 1996. http://www.w3.org/Search/9605-Indexing-Workshop/.

6. S. Garfinkel. *PGP – Pretty Good Privacy*. O'Reilly and Associates, Inc., 1995.

7. J. C. Reinbolt. License agreements on the Internet. In *California Computer Expo '96*, San Diego, Aug. 1996.

8. R. Rivest and B. Lampson. SDSI - A Simple Distributed Security Infrastructure. http://theory.lcs.mit.edu/~rivest/publ ications.html, Sept. 1996.

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

[4] `http://www.ncstrl.org/`