Repository in a Box Toolkit for Software and Resource Sharing

Shirley Browne [+]
Paul McMahan [+]
Scott Wells [+]

ABSTRACT

The Department of Defense High Performance Computing Modernization Program (HPCMP) seeks to bring technology to the warfighter. High performance software is being developed by the Common High-performance Software Support Initiative (CHSSI) component of the program. High-performance software and technology are being transferred to DoD users throughout the MSRC-supported Programming Environment and Training (PET) component. In any work environment, an individual's ability to perform his or her job is strengthened proportionally by the availability of the resources necessary to perform that job. The Repository in a Box (RIB) toolkit, developed by the University of Tennessee as part of the federally funded National High-performance Software Exchange (NHSE) project [1], provides mechanisms for sharing software and other resources among and between DoD Computational Technology Areas (CTAs) and sites, thus helping to bring this technology to the warfighter. RIB has been used to set up a distributed collection of interoperable software repositories for the DoD Major Shared Resource Centers (MSRCs). This network of repositories allows software, algorithms, and experiences to be shared within and between CTAs and MSRCs as well as with the broader DoD user community. Use of RIB provides a uniform and consistent user interface to these repositories. Extensions to the basic RIB data model are allowing value-added information, such as software evaluations, performance studies, deployment information, and intellectual property rights, to be cataloged. Thus RIB allows all pertinent information about the software to be consistently maintained and accessed from the same interface. Moreover, RIB enables DoD users to leverage other HPC efforts, such as NASA HPCC and NSF PACI, that are also using RIB to make resources developed by their programs available to users. This report gives an overview of the functionality of RIB, describes current DoD RIB repositories, and concludes with future plans.

[+] Computer Science Department, University of Tennessee, Knoxville

OVERVIEW OF RIB 1.0

Data Modeling Approach to Interoperability

The current RIB toolkit (version 1.0) provides the functionality to build and maintain a Web-based software catalog through the use of a file system based database to generate and display metadata about software and related entities. In order to categorize and describe software, certain naming conventions and classification mechanisms must be in place. This is handled through the use of the Basic Interoperability Data Model (BIDM) [2], an IEEE standard for software cataloging on the Internet. The BIDM provides a model for semantically defining classes, attributes, and relationships relevant to software. See Figures 1 and 2.

The BIDM is expressed in terms of an extended entity-relationship data model that defines classes for assets (the reusable software entities), the individual elements making up assets (i.e., files), libraries (i.e., repositories) that provide assets, and organizations that develop and manage libraries and assets.
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 BIDM specifies a minimal set of metadata that a software repository should provide about its assets in order to interoperate with other BIDM-compliant repositories. The basic model may be extended by defining additional subclasses for other relevant information.

Using the BIDM model, RIB provides HTML forms into which the instances of the data model can be entered. RIB then binds this metadata to HTML tags to create encodings that can be shared across the Web with other repositories that use the BIDM. Use of the BIDM is meant to ensure that metadata in one RIB repository employs the same semantics as that in another repository. Each of the classes, attributes, and relationships has well-defined semantics, which are specified in the BIDM document. While this may seem limiting, it is necessary to ensure the successful interoperation of RIB repositories. While the BIDM allows some flexibility in the amount of metadata detail, certain portions of the data model are required to provide a sufficient basis for interoperability.

To adequately organize software metadata in the visible catalog, a classification scheme has to be used for each repository. Such organization of software is currently handled by a domain (or classification) file, which is actually a simple HTML file that uses list tags to create headers for each software category appearing in the catalog. The information contained in this file is then parsed when a new software entry is to be added to the repository and each header is made available in a dropdown menu on the Asset metadata creation screen. Ideally, repositories wishing to interoperate with each other should have classification schemes that are closely related. That is, they should either mirror or complement each other.

What the end-user sees when visiting a repository created by RIB is a set of HTML pages that describe the software contained in that repository. These HTML pages are generated automatically by a set of administration scripts accessible to the repository maintainer via CGI, and their appearance can be customized to match the "look and feel" of the hosting web site. Access to the administration scripts for a repository (and to the contents of the repositories themselves if desired) can be protected by the usual HTTP server access control mechanisms. Each repository under a RIB installation can use its own set of access controls with separately configurable read and write access.

Although the BIDM has greatly enhanced the ability of software repositories to interoperate, it is desirable to be able to extend the basic model to cover specific areas more thoroughly or to meet specialized needs. One area for which an extension has already been defined is that of asset evaluation and certification.

Asset Certification Framework (ACF) [3] is an IEEE-standardized extension to the BIDM that defines a standard for the consistent structure, labeling, and description of evaluation and certification policies and results (see Figure 3). Most software repositories 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 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 repositories' evaluation and certification activities and results. By using the ACF, the repository maintainer can provide value-added information about evaluation of software according to general or CTA-specific evaluation criteria. Examples of repositories that utilize the ACF include the Parallel Tools Library (PTLIB) maintained by the NHSE project and the CEWES MSRC Grid Generation Software Catalog.

Intellectual Property Rights and Access Control

Recognizing the importance of intellectual property issues associated with software sharing, the IEEE has standardized another extension to the BIDM called the Intellectual Property Rights Framework (IPRF). The IPRF provides a model for incorporating rights issues into a repository's software metadata. Because rights of software may vary both within a single repository and between repositories, it is important to establish a common way to describe such intellectual property metadata associated with different assets. Issues such as software ownership, the type of intellectual property represented with the software (copyright, trademark, etc.), and the rights management policy of the entity controlling the software's distribution are all addressed within the IPRF model.

Recognizing the need to provide a mechanism for distributing software while maintaining access control, the NHSE project has developed the Access Control Toolkit (ACT) to provide file security as well as licensing mechanisms to manage software distribution. The ACT has a Web-based interface as well as a database backend and is currently in beta testing. Developed to make the job of technology transfer easier and more efficient, the ACT provides mechanisms to automate the software transfer process and reduce paperwork. When an end-user wishes to download a piece of software he fills out an electronic reggistration form which captures and places information about him into a database and supplies that information to the repository administrator. Depending on the release category of the software (e.g., freeware, commercial, etc.), an appropriate license is provided to the end-user. Upon verification of the user, the download request may then be forwarded to the appropriate tech transfer personnel who may approve or disapprove the request, or the user may already have been pre-approved to download certain categories of software. ACT includes mechanisms of varying strength such as email address verification, username/password pairs, and X.509 certification to verify the identity of the end-user. Figure 5 illustrates the internal mechanism of ACT.

An additional feature included in the Access Control Toolkit is a mechanism for allowing software uploads by software authors. The author is provided with an interface that enables him to provide technology transfer managers, and administrators with the necessary information for releasing the software. This feature reduces overhead and paperwork associated with performing the same task using conventional methods.

RIB Extensions for the DoD MSRCs

In addition to standardized extensions of the BIDM, organizations can develop their own extensions to meet specialized needs. During the development of repositories for the DoD MSRCs, the need arose for a mechanism for providing timely information about the deployment status of repository software on the MSRC platforms. To meet this need, the data model and functionality of RIB was extended to allow the creation of software deployment grids. A software deployment is an installation of a software package on a machine. The software deployment grid consolidates all of the software deployment information maintained by a repository into a concise matrix view that is automatically generated by RIB. By clicking on the entry in the matrix for a particular software package and a particular machine, the user can access site-specific usage information and tips as well as web-based tutorials. MSRC repositories that have made use of this software deployment capability include the CEWES and ARL programming tools repositories and the ASC and ARL CCM repositories. A portion of the CEWES tools deployment grid is shown in Figure 6.

Some of the application software cataloged in MSRC repositories is under active development, including work on performance optimization. To help support this activity, the RIB data model has been extended to manage application performance case studies. The data model for this extension is shown in Figure 7. The PerformanceStudy class contains a summary of the performance tuning methodology as well as results and points to more detailed information in the form of reports and papers and benchmark results. This extension is currently being used by the ARL PET team to document performance evaluation and optimization work on CCM, CFD, and other codes.

RIB VERSION 2.0

RIB Version 2.0 is completely rewritten in Java and makes use of the World Wide Web (W3C) Consortium's eXtensible Markup Language (XML) [4] standard and the Resource Description Framework (RDF) [5] recommendation to achieve maximum interoperability. XML is a new standard for structured documents that promises to revolutionize the way metadata are exchanged on the Web. XML is already supported by many Web applications, including the newest generations of Web browsers. Because RIB has adopted this standard, many other applications will be able to parse the data maintained by RIB. While XML provides a way to structure a document, the RDF provides a way to give semantic meaning to the elements in an XML document that is both human readable and machine processible. By using RDF and XML, the metadata generated by RIB will be usable by the broader Web user community.

The most substantial improvement in RIB version 2.0 is a totally revamped administration interface. In the previous version of RIB, the interface used HTML forms which could be cumbersome and tedious. At the time RIB 1.0 was implemented, Java and JavaScript was not used because of reservations about security and portability. Now that these technologies have matured considerably and become more acceptable, they have been embraced throughout the design of RIB and they provide a more flexible and intuitive interface. However, RIB 2.0 does not require Java or JavaScript in order to view the catalogs that it creates.

RIB version 2.0 will support a much greater variety of data models than did the previous version. While the BIDM will remain the default data model, RIB 2.0 will extend the capabilities of version 1.0 by allowing the repository administrator to add or delete entire classes from the data model. In effect, the repository administrator can stray completely from the BIDM and design a new data model from the ground up. This improvement makes RIB capable of managing practically any type of Internet-accessible resource. While this improvement opens the door to an entirely new range of applications for RIB, care must also be taken to prevent the proliferation of redundant data models. Interoperating repositories should follow the example set by the BIDM standardization effort and approach the data modeling stage of repository development with great care.

With RIB version 2.0, any attribute in a repository's data model will be able to use a controlled vocabulary. A controlled vocabulary is a predetermined set of terms that are used for the value of an attribute; any other values are not allowed. The previous version of RIB used a controlled vocabulary for the Domain attribute in the Asset class. Just as the Domain attribute was used to sort catalogs in RIB 1.0, any attribute that uses a controlled vocabulary will be usable for sorting catalogs in RIB 2.0. The sorting attribute will be adjustable dynamically, thus allowing the creation of many different organizations of catalogs from the same set of information.

The flat file database underlying RIB 1.0 limited the ability to make sweeping changes to large collections of data without a great deal of overhead. RIB version 2.0 interfaces to a database backend to make data management much more reliable and straightforward. SQL commands are used by RIB to perform updates to large numbers of records in an optimized fashion. Using database concurrency control helps prevent race conditions in multi-user environments where the same data might be modified simultaneously by different users. A database makes RIB's data more portable in that the data can be moved to different locations without changes. The data can also be easily exported to other applications via standard SQL. Figures 8 and 9 illustrate the data management functions of RIB 1.0 and RIB 2.0, respectively.

Many users of RIB version 1.0 asked for a feature that would enable them to add new data to a holding area before it appears in their HTML catalog. This feature will enable them to accept input from sources not deemed completely reliable, and to check that input before incorporating it into their catalog. This feature is included in RIB 2.0 in the form of an "object approval" operation. Whenever data is entered into the RIB system, it is marked as "not approved" by default. Until the data has been explicitly approved, it will not appear in the catalog, nor will it be available for interoperation. This feature can be disabled if it is not needed.

Another improvement in RIB 2.0 is its interoperation facilities. Previously, interoperation was only possible on a per-object basis. This meant that in order for repositories to interoperate, they had to manage individual pointers to each other's data objects. This fine-grained level of interoperation was difficult to manage because it forced administrators to constantly monitor each other's collections and to manually add or remove pointers whenever remote objects were added or deleted. Interoperation becomes much easier to manage on a per-repository basis because repositories that wish to interoperate need only retain a pointer to each other's repositories. Each repository can dynamically generate a list of its objects and when they were last modified. Other repositories poll this list and updaate their collections whenever necessary. Figures 9 and 10 illustrate the differences in the interoperation feature between versions 1.0 and 2.0, respectively.

The HTML catalog generation feature in RIB version 2.0 also includes many improvements. The table of contents for the catalog is now organized in a fashion that Web users are more familiar with - instead of listing the entire table of contents on one page, it is categorized into separate pages. This improvement eliminates the cluttered look that was apparent in larger RIB 1.0 catalogs. A "What's New" feature has been added to the catalog to automatically track new additions to the repository and allow users to query this listing based on the number of days that they wish to look back. Another improvement to the catalog is greater flexibility in the views that may be constructed from the underlying database. RIB 1.0 always generated catalogs for the Asset class, sorting by the Domain attribute. With 2.0, catalogs can be generated based on any class and sorted by any attribute that contains a controlled vocabulary. Management of the HTML catalog is now easier because the administrator does not need to ask RIB to regenerate the repository's table of contents after making changes to the database. Since all HTML pages are dynamically generated, any changes made to a repository are immediately visible in the repository catalog, provided approval has been granted via the object approval function.

The software deployment grid feature that was added to RIB 1.0 is an example of a join of the Asset and Machine classes with the intersection of those two classes being the deployment of an asset on a machine. This feature has been generalized in RIB 2.0 to allow arbitrary selections and joins to be generated from the underlying database and displayed.

The password system used to prevent unauthorized access to a repository's administration pages was not completely straightforward in RIB 1.0. The HTTP server, upon which RIB depends, had to be manually reconfigured to enable security controls. However, in RIB 2.0 the access control is managed directly in the RIB administration interface.

CONCLUSIONS AND FUTURE WORK

RIB has already been used to set up a number of software repositories for the DoD MSRCs. These include repositories for Programming Tools (ARL, ASC, and CEWES), CCM (ARL and ASC), CFD (ARL and CEWES), SIP (ARL), and Grid Generation Software (CEWES). The CTA-specific software repositories contain information about application area software and tools, including commercial and research software. The Grid Generation software repository provides information about grid generation software of potential value to the CFD, CSM, CWO, and EQM CTAs. The Programming Tools repositories provide information, including deployment status, about parallel programming tools such as debuggers, performance analyzers and parallel I/O systems. In addition to its use at the DoD MSRCs, RIB has been used to set up software repositories for the NASA HPCC program and the NSF PACI sites. For an up-to-date list of current RIB software repositories, see the NHSE RIB home page [6]. Similarly to how a number of different discipline-oriented repositories have been pulled together into the top-level NHSE software catalog that can be searched and browsed from one place (as shown in Figure 12), the DoD MSRC repositories can be pulled together into one top-level catalog. In addition, RIB can be used to provide a "corporate memory" of the CHSSI program, similar to how the NASA HPCC repository has used RIB to keep a record of software produced by funded projects as well as when and how they have met their performance milestones.

In the future, we envision RIB being used as the basis for development of high-level, problem solving environments (PSEs), both application-area specific PSEs as well as cross-cutting programming tools and visualization PSEs, for which RIB will provide the underlying database of available resources and tools. As an example, consider how a PSE might assist users of a programming tools repository. Rather than requiring the user to locate and read deployment and usage information about a performance analysis tool, the PSE could provide a graphical user interface from which the user could select and invoke the tool. In addition, the PSE could manage performance data generated by the tool and automatically generate metadata relating the performance data to the particular software and hardware environment that produced it for insertion into a performance case study. PSE interoperation could be greatly enhanced by applying the same extensible but standards-based data modeling approach used in RIB. Such interoperation would facilitate coupling of software and tools from different PSEs, for example coupling a debugger with a data visualization tool, or executing an application from a CTA PSE under the control of a debugger.

ACKNOWLEDGEMENTS

This research was partially funded by the ARL, ASC, and CEWES Major Shared Resource Centers of the DoD High Performance Computing Modernization Program through the Programming Environment and Training (PET) initiative. Views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of Defense position, policy or decision unless so designated by other official documentation.

REFERENCES

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

[2] Institute of Electrical and Electronics Engineers (IEEE), *IEEE Standard for Information Technology-Software Reuse-Data Model for Reuse Library Interoperability: Basic Interoperability Data Model (BIDM)*. Std 1420.1 - 1995.

[3] Institute of Electrical and Electronics Engineers (IEEE), *IEEE Guide for Information Technology*--*Software Reuse-Data Model for Reuse Library Interoperability: Asset Certification Framework (ACF)*. Std 1420.1a-1996.

[4] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, eds. *Xtensible Markup Language (XML) 1.0*. World Wide Web Consortium, (http://www.w3.org/TR/1998/REC-xml-19980210). January 1998.

[5] Ora Lasilla and Ralph R. Swick, eds. *Resource Description Framework (RDF) and Model and Syntax Specification Recommendation*, World Wide Web Consortium, (http://www.w3.org/TR/PR-rdf-syntax). January 1999.

[6] http://www.nhse.org/RIB/

FIGURES

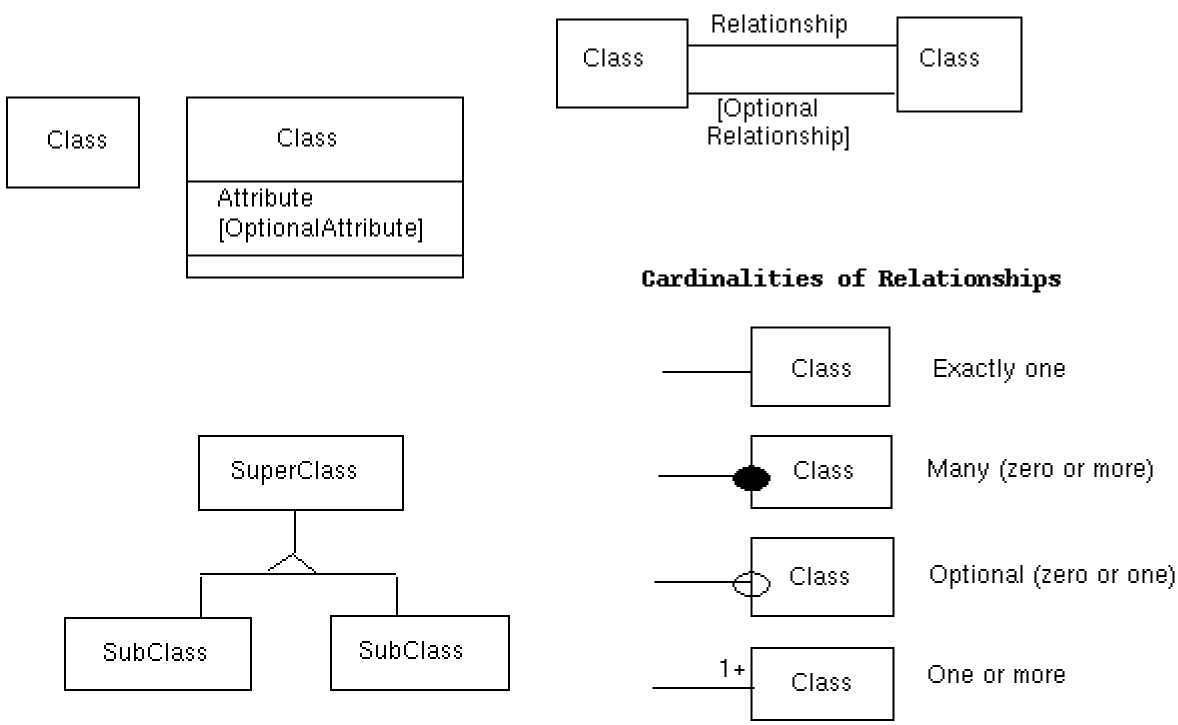

Figure 1. Basic Interoperability Data Model (BIDM) legend

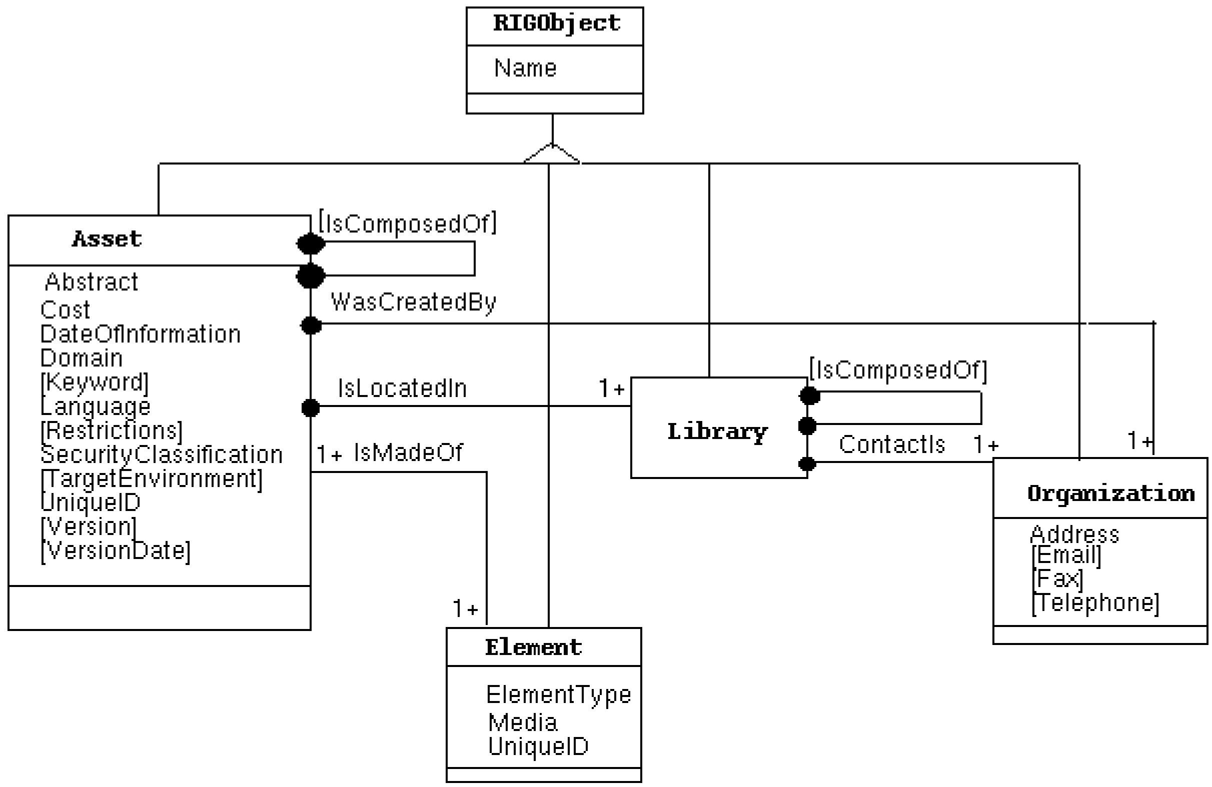

Figure 2. Basic Interoperability Data Model

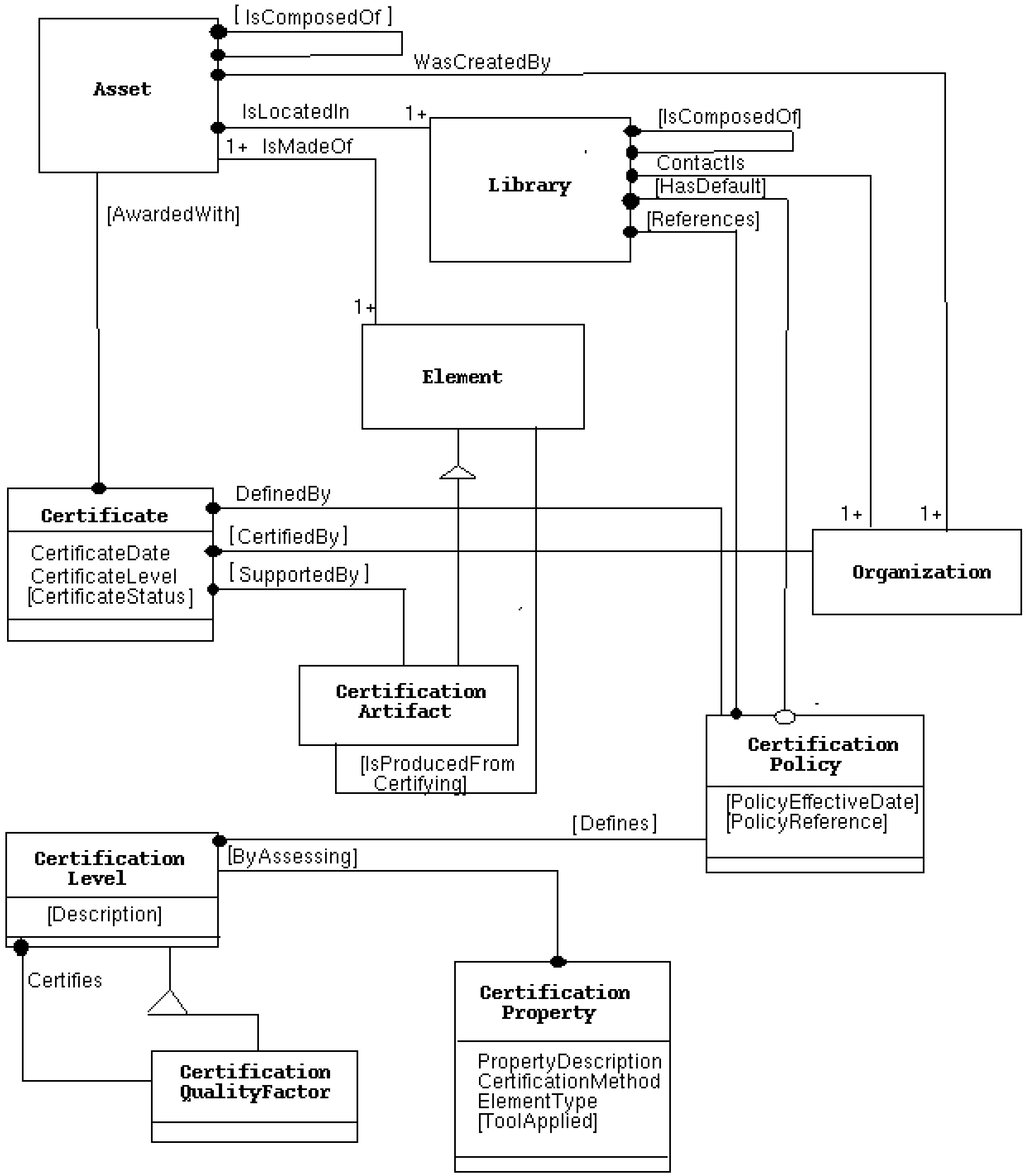

Figure 3. Asset Certification Framework (ACF)

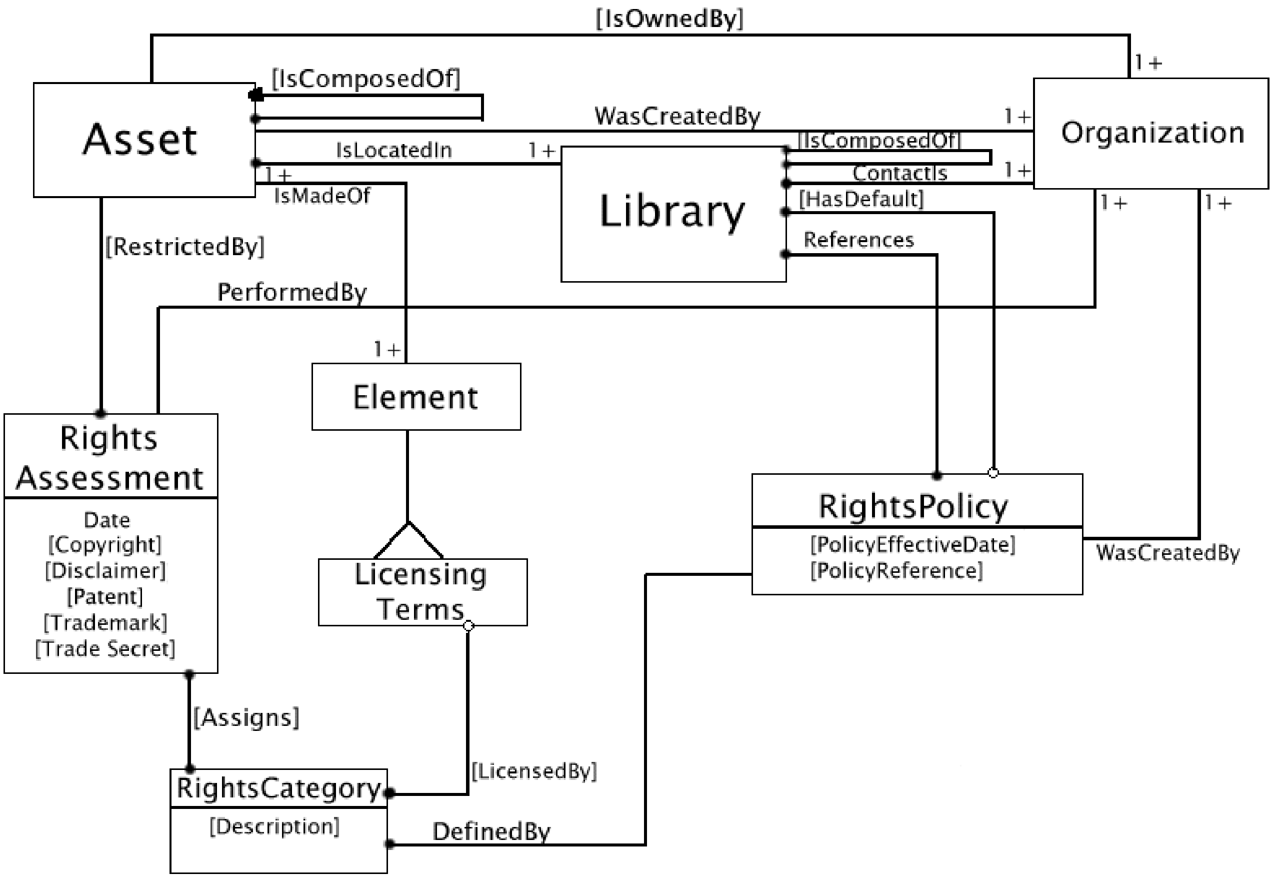

Figure 4. Intellectual Proprerty Rights Framework (IPRF)

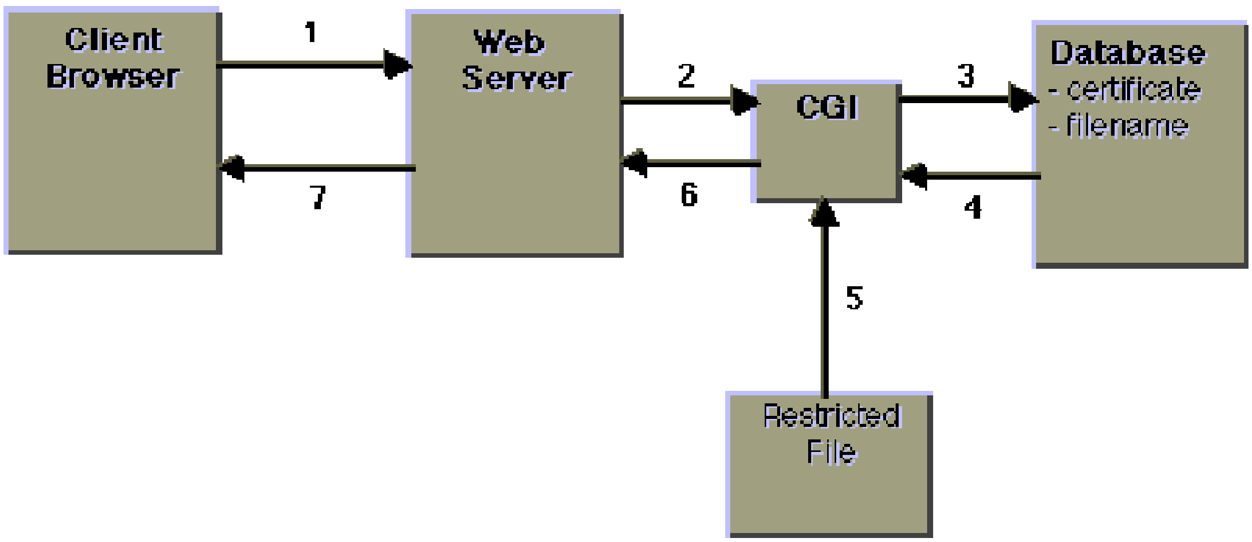

Figure 5. Access Control Toolkit operation

	Cray T3E (jim)	IBM SP (osprey)	IBM SP (pandion)	SGI/Cray Origin 2000 (origin)
Cray Scientific Library	installed	-	-	-
Cray TotalView	installed	-	-	-
Etnus TotalView	-	installed	installed	installed
IBM ESSL 2.2	-	installed	installed	-
IBM PESSL 1.2	-	installed	installed	-
IBM PIOFS	-	installed	-	-
Java Development Kit	-	installed	installed	installed
LAPACK	-	-	-	installed
MPP Apprentice	installed	-	-	-
OpenMP	-	-	-	installed
PAT	installed	-	-	-
SCSL	-	-	-	installed
SGI CHALLENGEcomplib	-	-	-	installed
SGI Parallel Analyzer (cvpav)	-	-	-	installed

Figure 6. A Portion of the CEWES SPP Tools Deployment Grid

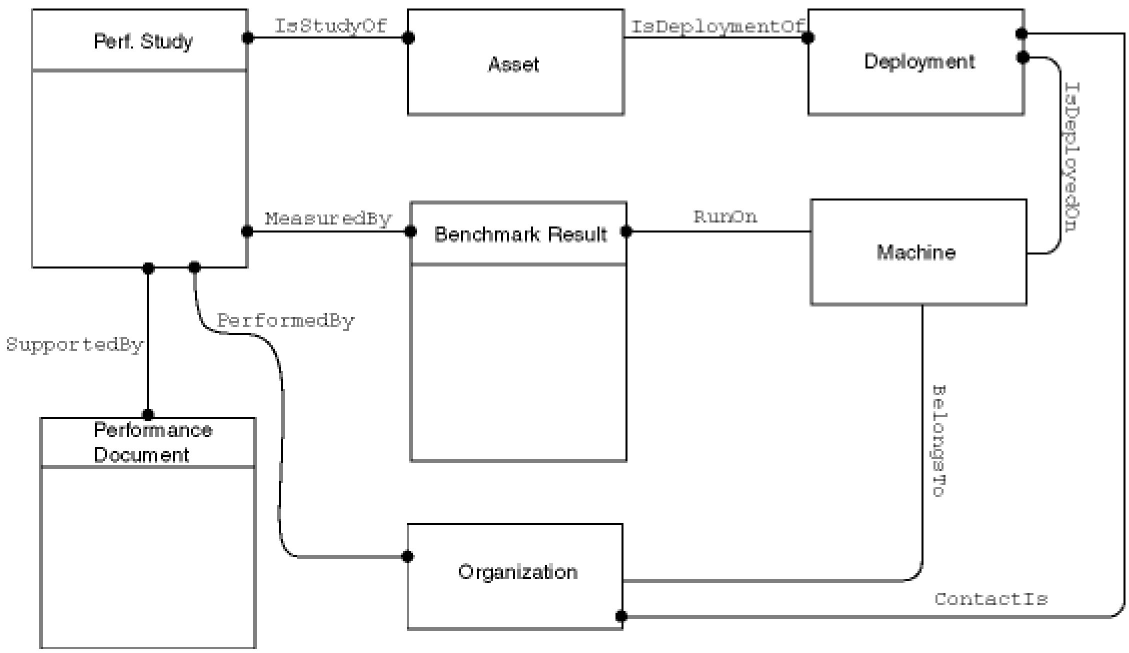

Figure 7. Performance Case Studies data model

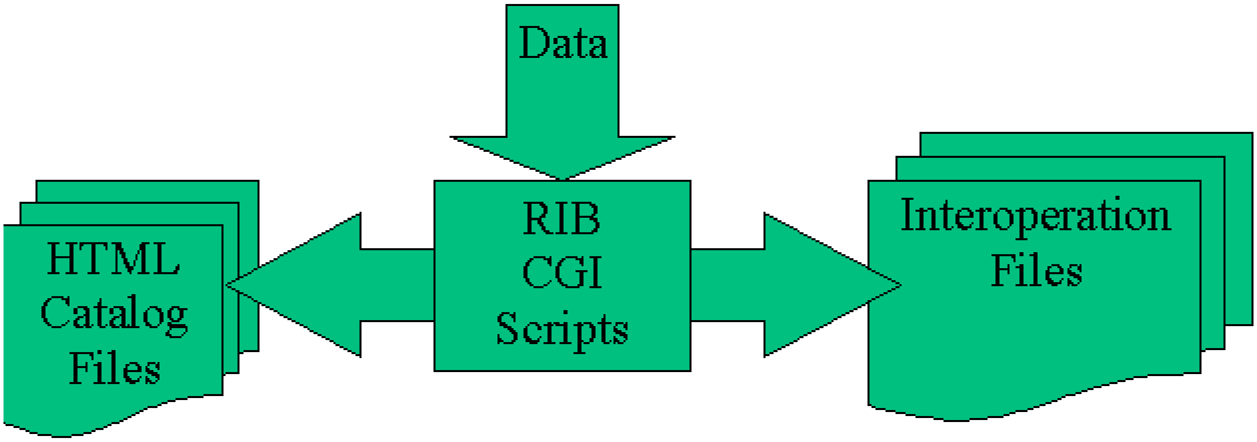

Figure 8. Data Management in RIB 1.x

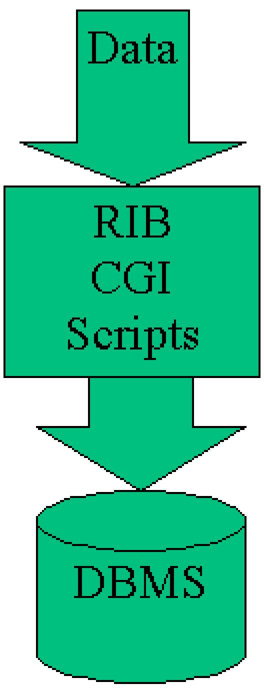

Figure 9. Data Management in RIB 2.x

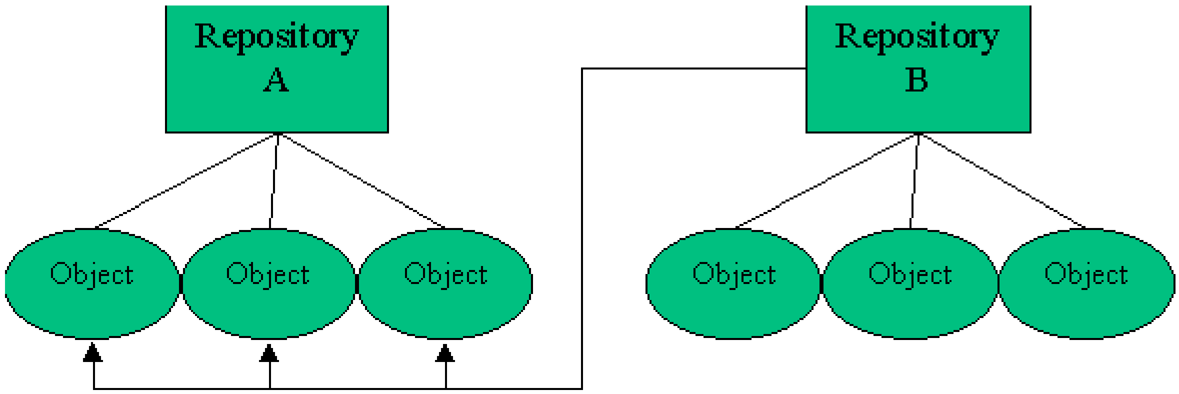

Figure 10. Interoperation in RIB 1.x

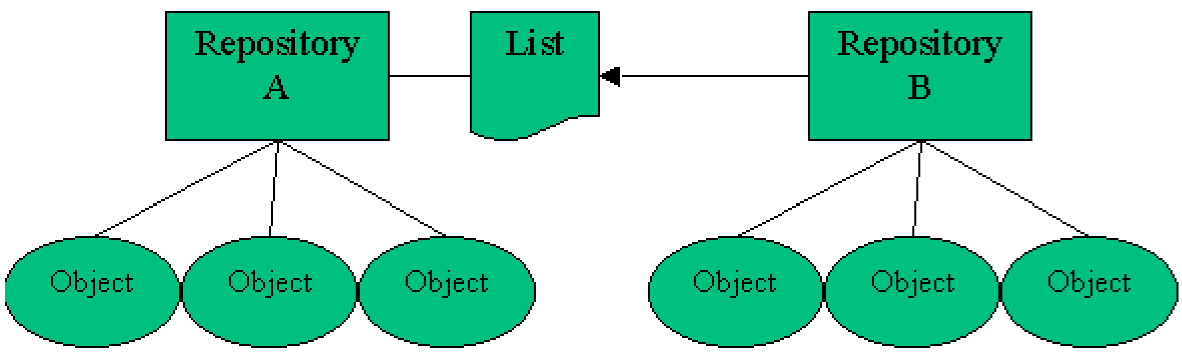

Figure 11. Interoperation in RIB 2.x

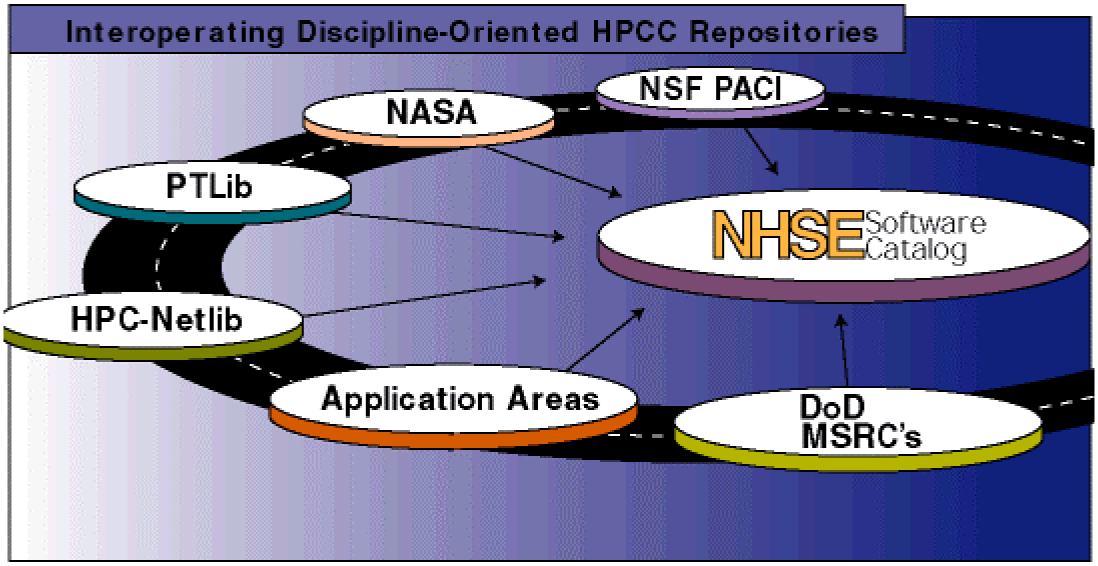

Figure 12. Top level interoperation