2000

A Survey of Software Reuse Repositories

Guo, Jiang; Luqi

A Survey of Software Reuse Repositories*

Jiang Guo
Research Associate
US National Research Council
NPS/CS, Monterey, CA 93943, USA
Gj@cs.nps.navy.mil

Luqi
Department of Computer Science
US Naval Postgraduate School
Monterey, CA 93943, USA
Luqi@cs.nps.navy.mil

Abstract

Reuse libraries are organizations of personnel, procedures, tools, and software components directed toward facilitating software component reuse to meet specific cost-effectiveness and productivity goals. The paper gives a survey of the major software reusable component repositories. This survey will be a base to develop future efficiently searchable, user-friendly, useful, and well-organized repositories.

1. Introduction

Reuse libraries are directed toward facilitating software life cycle component reuse to meet cost-effectiveness and productivity goals [1]. The principal rationale for the existence of a reuse library is to provide ready access to reusable components by the staff of development and maintenance organizations, and to support system composition and rapid prototyping [2, 3]. The number of cases in which library systems are successfully being used to maintain code and other reusable software life cycle components continues to increase. It is essential that the library system support developers and other users in the process of locating, retrieving, comparing, and maintaining reusable software components.

Reuse libraries are only one critical element of successful reuse program. In the past, reuse has primarily been the result of opportunistic success, where one program was able to take advantage of the efforts of another. There must be a paradigm shift from current software engineering and development practices to a software engineering process in which software reuse is institutionalized and becomes an inseparable part of the software development process. Reuse must be systematic, driven by a demand for software components identified as a result of domain analysis and architecture development. Reuse needs to be treated as an integral part of engineering and acquisition activities. Most importantly, it is essential that an organizational infrastructure be implemented to manage domains, define products and standards, establish ownership criteria, allocate investment resources, and direct the establishment and population of reuse libraries. An effective infrastructure will guide reuse activities to avoid duplication of effort, impose necessary standardization, and ensure library population is user demand-driven.

2. Library Mechanism

Usually, critical reuse library capabilities include the following:

- automated library system with a Graphical User Interface, for browsing, searching and retrieval;
- standard component framework (e.g., to include purpose, functional description, certification level, key environmental constraints, historical results of usage and legal restrictions);
- effective classification scheme for each domain; and,
- thorough system and component documentation.

Each library system must be designed to provide as much automated support as possible to users in identification, comparison, evaluation, and retrieval of similar reusable components. Support for adapting, transforming, and specializing components is desirable. It must also provide a range of support to users in locating and comparing the relative reusability of individual library components. Furthermore, the system must be readily available to system developers if it is to be used, and must support access from a variety of platforms. As the library acquires significant number of Reusable Software Components (RSCs), an automated search and retrieval system becomes indispensable [4, 5, 6]. Whatever tool is used, the library must have a way to classify RSCs so that a user can quickly find what is wanted without frustration and delay. Sophisticated, expert system, knowledge-based approaches and new technologies for high-speed text search are the subjects of current research efforts. Generally speaking, software reusable component retrieval

* This research was supported by ARO(38690-MA) and DARPA(99-F759).

methods include browsing, keyword searching, facet approach, syntactic matching, and semantic matching [1].

Standard component frameworks help ease the process of comprehension and comparison of similar components, and include data such as relative numeric measures for reusability, reliability, maintainability and portability [7]. Inclusion of testing and component documentation provides additional information to help the potential user gauge the effort required to tailor the component for reuse.

Effective classification schemes are essential to assist the user in locating and comparing library components, and to speed the process of identifying appropriate components for the task at hand [8, 9]. Finally, system and component documentation complete the cycle of evaluation, and enable the reuser to determine which components have reuse potential with regard to specific requirements, and to fully comprehend the process of obtaining components for reuse in a new application.

In addition, other equally important requirements have been identified that require resolution in order to support cohesive, wide reuse. These include l) integration of library capabilities and procedures within the system development and acquisition process; 2) identification and support of specific requirements associated with the security and integrity of reusable components implementing Trusted Computing Base (TCB) or other security capabilities; and 3) intercommunication and interoperabllity among diverse library systems. Experience has shown that these requirements can only be resolved through the combination of developing technology, standard procedures and evolution or revision of existing policies.

There are different communities for which a repository is necessary, and each community has somewhat different repository requirements. These communities include the national or horizontal communities; the local or, internal communities, and a number of domain-specific vertical communities [10].

3. Library Operation

The reuse library, while essential, is but one ingredient in a successful reuse program. Experience has shown that actual support of reuse activities within a target domain must include a range of programmatic and technological support that includes domain analysis activities, user indoctrination and training, metrics collection and analysis, reuse engineering support, and component certification and reengineering.

The importance of domain analysis activities as an initial step in implementation of a reuse library cannot be over-emphasized. Domain analysis activities are considered to be an integral part of providing reuse support to various programs. Standard products of domain analysis include identification of high-demand categories of reusable components, domain-specific models and architectures, and domain specifications and taxonomies. These direct products also provide the basis for development of long-term implementation plans and domain knowledge bases.

In order to measure reuse success, the library must collect and analyze considerable data in a continuing assessment of the library's procedures and tools, the usefulness of its RSC collection, the accuracy of RSC classifications, and the general responsiveness of the library to the needs of users.

The library staff receives direction in the form of specific operational objectives, principally aimed at making software reuse cost-effective. In addition to ensuring that RSCs are available, the library is in a position to provide other support to help ensure that the benefits of reuse are realized, including the distribution of published manuals like Standards and Guidelines and user documentation for library tools. In addition, on-call assistance should be made available to users. Reuse engineering support encompasses a wide range of engineering activity. These activities will include working within individual system development and maintenance efforts to assist in (l) identification, selection and reapplicatlon of existing reusable software components, (2) quantification of potential savings or cost avoidance as a result of reuse, and (3) design and implementation of software products that will themselves be reusable in future efforts.

Another key area is thorough library system documents. Documentation has proven to be an essential aspect in establishing and operating a library.

4. Some Reusable Software Component Repositories

4.1. Commercial Repositories

- **+1Reuse Repository**

The +1Reuse system was developed by +1 Software Engineering Co. in California [31]. It is now running on Sun Workstation platforms. Operating system is Solaris. GUI is based on OpenWindows, Motif, and CDE.

The +1Reuse system supports reuse repositories created and maintained by the user, project-wide "filtered" repositories under strict quality controls, and selective reuse. Selective reuse enables reuse of any submodel from an existing or re-engineered +1Environment project. In a sense, every +1Environment project is a reuse library. Selective reuse significantly improves a user’s ability to reuse all source code and documentation from all previous projects and at any granularity. (To the best of our knowledge, they are the only company to support this feature.)

The +1Reuse system supports reuse of: design, documentation, source code, header files, test cases, test shell scripts, expected test results, and modeling information.

All source code reversed engineered or developed using the +1Environment can be reused. +1Reuse addresses reuse issues such as reuse of source code under configuration management and duplicate file names. +1Reuse supports three forms of reuse: User-Defined Reuse Library, Filtered Reuse Library, and Selective Reuse. Since a programmer's productivity can be increased by reusing existing code and documentation, +1Reuse helps to make all source code, documentation, header files, and test files reusable by its support of submodels. After a submodel has been selected, +1Reuse copies the submodel and its associated files to the new project and helps to resolve a number of problems which may arise (e.g., identical file names and files checked in under configuration management).

- **Software Asset Library Management System (SALMS)**

SALMS is a system for classifying, describing, and querying reusable assets [32]. Reuse of software assets at all phases of the software engineering life-cycle is recognized as being one of the major enablers for productivity and quality improvements. However, a common inhibitor to company-wide reuse is often the lack of visibility of reusable assets within the developer community.

A central repository for reusable assets provides a solution to this problem. The main purpose of such repository is to provide mechanisms for classification and storage of software assets, along with techniques for efficiently retrieving them.

SALMS (Software Asset Library Management System) is a software product which provide these mechanisms. It fills the gap between development for-reuse activities (building, acquiring, or re-engineering of reusable assets) and the development with-reuse activities (using reusable assets in the creation of new software products). It plays a central role in the implementation of a company's reuse program.

In addition, SALMS also provides features for the requirement management activity, and for the creation and management of a company's technical library. SALMS can be distributed over customer's network of PCs or UNIX workstations and thus be accessible by all developers within a software organization. The user interface is based on WEB Technology.

In SALMS, an asset can be viewed as a collection of artifacts produced throughout the life-cycle, such as requirements, architecture models, design specifications, source code, or test scripts.

- **Automated Software Reuse Repository (ASRR)**

The Automated Software Reuse Repository (ASRR) tool provides users with a searchable repository of reuse information [33]. It consists of two main parts, the administration tool and the reuse repository. The administration portion of the tool performs user administrative functionality including: the ability to add, delete, or change users and their attributes. The attributes include the following: security levels, group and security permissions to add, edit and delete modules. The reuse repository allows the user to upload modules and store them in a searchable repository.

The ASRR provides the following functions:

- Program Control. Provides complete login control for the ASRR.
- Protection. The ASRR can limit a user's edit, delete, viewing, add, upload and download module permissions through the administration portion of the tool.
- Security. The ASRR tool provides extra security for inactive users by logging them out of the ASRR after a 30-minute period of inactivity.
- Easy Access to Reuse Items. The ASRR tool allows registered users flexibility in searching for reuse items in the reuse repository by allowing the users to search for strings of words using "not", "or", or "and" in searching.
- Reuse Information Readily Available for Users. Specific information is available for reuse module items including the platforms utilized, ease of reuse and any additional information obtained from users.

- **The Universal Repository**

The Universal Repository was developed by Unisys [34]. It is designed to help customers move forward into a repository-based development environment.

The Universal Repository, which is based on object-oriented principles, can function as the backbone of a flexible workgroup or enterprise development environment. At the core of this repository is the Repository Services Model (RSM) - which can encompass representations of all tools, database management systems (DBMSs), programming languages, business rules, and data.

Customers can extend the Universal Repository by adding their own models based on the structures provided in the RSM. The summation of all models defined in a repository is called the information model. Each part of customers' development environment becomes an integrated piece of the whole when customers use the models encompassed within the information model. This unified view enables both developers and customers to achieve inter-tool integration.

In addition to its modeling capabilities, the Universal Repository offers features that enhance customers' development environment, manage organizational information, and make such information available to everyone in a customers' organization.

Unisys is dedicated to improving customers' product lines with the Universal Repository. Support and training are available to help customers quickly adopt this new technology. By providing a shared catalog of all software components, a repository promotes reuse. It makes it easy to locate and access components for reuse in multiple applications. Reusing software components can enhance quality. Customers can develop, validate, and verify a component for use in one product. When customers reuse that component, they expend less time and fewer resources to validate and verify that component for use in other products [11]. A single change to correct a defect in a reused component is reflected in all tools using that component. Such consistency among products ensures their integration and interoperability when you port them to different operating platforms.

- **AIRS**

AIRS is an AI-based library system for software reuse, which was developed by E.J. Ostertag, J.A. Hendler, R. Prieto-Diaz, C. Braun [12]. AIRS allows a developer to browse a software library in search of components that best meet some stated requirement. A component is described by a set of (feature,term) pairs. A feature represents a classification criterion, and is defined by a set of related terms [10, 12]. AIRS also allows representation of packages, that is, logical units that group a set of related components. As with components, packages are described in terms of features. Unlike components, a package description includes a set of member components. Candidate reuse components (and packages) are selected from the library based on the degree of similarity between their descriptions and a given target description [13]. Similarity is quantified by a non-negative magnitude (called distance) that represents the expected effort required to obtain the target given a candidate. Distances are computed by functions called comparators. Three such functions are presented: subsumption, closeness, and package comparators. The AIRS classification approach is based on a formalization of the concepts and is similar to faceted classification [44]. The functionality of a prototype implementation of the AIRS system is illustrated by application to two different software libraries: a set of Ada packages for data structure manipulation, and a set of C components for use in Command, Control, and Information Systems.

- **Reuse Library Toolset (RLT)**

EVB Software Engineering, Inc. announced the commercial release of the Reuse Library Toolset (RLT) in 1994 [35]. RLT is a system for creating and managing collections of reusable assets independent of programming language, design method, or development process. To represent all life-cycle assets RLT employs the Extended Faceted Classification System, controlled keyword, attribute value (frames), and asset interdependencies.

Experience has shown that the cost of producing software is significantly reduced when reuse is an integral part of the process. RLT supports all reuse oriented tasks, from library management through domain analysis to asset search and retrieval. With its intuitive graphical user interface, RLT is easy for beginners to learn, yet provides powerful functionality for advanced users with complex needs.

RLT provides reuse and library metrics, client-server architecture, and ability to exchange library information across multiple platforms and databases. These include: DEC Alpha OSF1, HP/UX, SGI, SunOS, Solaris, Informix, Oracle, and Sybase. Additional platforms have been supported in 1995 include: Windows 3.1/NT and OS/2.

RLT's open architecture allows easy integration with existing CASE and development tools, such as structure design tools, versioning systems and configuration management systems.

- **HSTX Reuse Repository**

The HSTX Reuse Repository was developed by Hughes STX Corporation [36]. The mechanisms are designed so users can search/browse the contents of the Reuse Repository for what they need and submit contributions to the Reuse Repository librarian through WWW pages.

4.2. Government Repositories

- **Defense Software Repository System (DSRS)**

The DSRS is an automated repository for storing and retrieving Reusable Software Assets (RSAs) [14]. The DSRS software now manages inventories of reusable assets at seven software reuse support centers (SRSCs). The DSRS serves as a central collection point for quality RSAs, and facilitates software reuse by offering developers the opportunity to match their requirements with existing software products.

DSRS accounts are available for Government employees and contractor personnel currently supporting Government projects. The Account Request Form must be approved and signed by the requestor's Government Project Manager prior to submission to the SRP. The Customer Assistance Office (CAO) is the SRP point of

contact for both technical and non-technical information and support.

The Defense Software Repository System (DSRS) supports reusable asset classification to comply with published guidance (DoD 8020.1-M and TAFIM), support domain engineering, establish more effective asset searching, and increase interoperability. The DoD software community is trying to change its software engineering model from its current software cycle to a process-driven, domain-specific, architecture-based, repository-assisted way of constructing software [15]. In this changing environment, the DSRS has the highest potential to become the DoD standard reuse repository because it is the only existing deployed, operational repository with multiple interoperable locations across DoD. Seven DSRS locations support nearly 1,000 users and list nearly 9,000 reusable assets. The DISA DSRS alone lists 3,880 reusable assets and has 400 user accounts.

DSRS is adaptable to additional types of reusable assets and better methods of describing them. The description of repository assets is called classification. This paper reports the results and recommendations of a study of classification methods for storage and retrieval of Reusable Assets (RAS) in the DSRS. The Defense Software Repository System (DSRS) reusable asset classification is changing to achieve policy compliance, support domain engineering, establish more effective asset searching, and increase interoperability.

The far-term strategy of the DSRS is to support a virtual repository. These interconnected repositories will provide the ability to locate and share reusable components across domains and among the services. An effective and evolving DSRS is a central requirement to the success of the DoD software reuse initiative. Evolving DoD repository requirements demand that DISA continue to have an operational DSRS site to support testing in an actual repository operation and to support DoD users. The classification process for the DSRS is a basic technology for providing customer support [16]. This process is the first step in making reusable assets available for implementing the functional and technical migration strategies.

- **Library Interoperability Demonstration (LID)**

The Library Interoperability Demonstration (LID) is a prototype library system [17, 18]. It is used to illustrate how monolithic reuse libraries can be decomposed into distinct, functional layers connected by open interfaces, such as those specified by Asset Library Open Architecture Framework (ALOAF). It is a collaboration between SAIC and Unisys. The demonstration shows how the physical storage of assets can be separated from the cataloging of assets, and how a user can choose a single, local, user interface tool to access multiple reuse libraries.

The STARS Program developed a specification of an ALOAF to support an "open systems" approach to constructing asset libraries. The ALOAF evolved to incorporate interfaces specifically intended for interoperability, culminating in the release of ALOAF Version 1.2 [19]. The LID builds upon the open interfaces provided by ALOAF, its associated Asset Interchange Language (AIL), PCTE, OSF/Motif, and POSIX. As shown in the LID Software Architecture diagram, a reuse library can be divided into three distinct layers which are connected via open interfaces, thus providing opportunities for interoperability at each layer. The three layers are:

- User Interfaces. The demonstration includes two user interface tools: a graphical browser derived from the Unisys Reuse Library Framework (RLF) and a text-based browser modeled after SPS's InQuisiX reuse library system. Both tools are built upon Ada bindings to OSF/Motif.
- Asset Catalogs. The demonstration shows two asset catalogs. The first catalog is derived from the Unisys collection of ASW components, and resides on an IBM RISC System/6000 at the STARS Technology Center. The second catalog is derived from SAIC's collection of flight simulator components, and resides on an IBM RISC System/6000 at the SAIC offices in Orlando, FL. The interface between each of the catalogs and the user interface tools is defined by the ALOAF.
- Asset Storage. In the demonstration, the storage of assets is provided by the AFS cell at the STARS Technology Center. Neither catalog stores assets itself; instead, both catalogs "subcontract" the storage function to the AFS server.

- **Integrated - Computer Aided Software Engineering (I-CASE)**

I-CASE was developed by Air Force Reuse Center (AFRC) [38]. The Air Force Reuse Center is the Air Force Management Information Systems (MIS) repository for reusable software assets. These assets are primarily Ada source code modules consisting of Government and commercial packages. The library has over 1,200 assets including many assets of the system life-cycle, such as requirements, designs, documentation and source code. Integrated Computer-Aided Software Engineering (I-CASE) provides a contract for DoD users to purchase an integrated set of tools that will automate many of the MIS software development activities over the entire software development and maintenance life-cycle. I-CASE also provides the support elements necessary to implement, operate, and maintain the I-CASE environment (i.e., training, maintenance, and technical support). The overall strategy of this project is to automate reuse processes through an Integrated-Computer Aided Software Engineering (I-CASE) environment. The specific strategy

is to implement these reuse processes within a workflow environment to certify or re-engineer reusable assets as quickly as possible. The EVB Reuse Library Tool (RLT), supplied as part of I-CASE, is used as the reuse repository tool.

- **Multimedia Oriented Repository Environment (MORE)**

As the World Wide Web (WWW) becomes very popular among internet users, an increasing number of public repositories are using the WWW to promote their services. The Electronic Library Services and Applications (ELSA) project is the operational part of the Repository Based Software Engineering (RBSE) program [20]. RBSE is a National Aeronautics and Space Administration (NASA) sponsored program dedicated to introducing and supporting common, effective approaches to designing, building, and maintaining software systems by using existing software assets stored in a specialized library or repository.

In addition to operating a software lifecycle repository, RBSE promotes software engineering technology transfer, academic and instructional support for reuse programs, the use of common software engineering standards and practices, software reuse technology research, and interoperability between reuse libraries/repositories.

During its life cycle, the ELSA project responded to emerging technologies, the growing sophistication of its client base, and industry trends by advancing the capabilities of its management software. Thus, ELSA stands as a customer-driven environment employing an advanced library management mechanism, MORE (Multimedia Oriented Repository Environment).

ELSA replaced AdaNet on August 31, 1994 when the first public access to its new service was granted. The library is the operational part of the Repository Based Software Engineering (RBSE) program which is a NASA sponsored initiative in software reuse. In a timeframe of approximately two weeks, ELSA transitioned its library holdings and accompanying metadata from a monolithic X-Windows based system to MORE. The improved interface employs client/server technology and is accessible through the WWW. MORE is a public domain, metadata based repository tool employing the WWW as its sole user interface. It consists of a set of application programs which operate together with a stock httpd server to provide access to a database of metadata [21]. The entire interface, client browsing and searching, repository definition, data entry and other administrative functions, are provided through stock Web clients.

Repository assets are classified using a collection (topic) and class (type) paradigm. According to their subject matter, they are included in the collections or subordinate collections that best represent domain coverage. The assets are also classified by media or information type through the class approach. Thus, users can view the information from a top-down perspective through the hierarchy of collections or across collections by the hierarchy of classes.

MORE was designed to support this collection and class model. Navigation is achieved through the activation of high-level hypertext links which ultimately lead to metadata or assets themselves. Searching (Natural Language or Pattern Match) is performed against information provided in the metadata [22, 23, 24]. This combination provides users with a reliable and efficient means of accessing a high volume of assets.

Administrative functions are specifically designed to meet librarians' needs. For instance, assets are stored in "developmental" mode which provides a cleanroom environment for the performance of population and/or certification activities. Developmental assets are only available for viewing by librarians. Following the completion of these processes, each asset is promoted to "production" mode and is therefore accessible to the general user population.

Each collection can have one or more groups associated with it that are authorized to access the assets and subcollections making up the collection. Groups in turn are made up of sets of users and other groups; all defined through the librarian interface. Users not transitively a member of a designated group for a given collection will never see the collection, or its contents, through any of the browser or search mechanisms. This mechanism supports the definition of multiple virtual repositories in a single physical repository, reducing administrative overhead and allowing direct sharing of assets.

- **Asset Source for Software Engineering Technology (SAIC/ASSET)**

Asset Source for Software Engineering Technology (SAIC/ASSET) offers products and services in digital library support, electronic commerce and software engineering with an emphasis on reengineering and reuse [26]. SAIC/ASSET, established by Advanced Research Projects Agency (ARPA) as a subtask under the Software Technology for Reliable Systems (STARS) program, is transitioning to a private enterprise as a division of Science Applications International Corporation (SAIC).

SAIC/ASSET's primary mission is to provide a distributed support system for software reuse with the Department of Defense (DoD) and to help foster a software reuse industry within the United States. SAIC/ASSET's initial and current focus is on software development tools, reusable components and documents on software development methods. SAIC/ASSET is participating in interoperation with other reuse libraries such as:

- Comprehensive Approach for Reusable Defense Software (CARDS)
- Ada and Software Reuse Information Clearinghouse Defense Software Repository System (DSRS)
- Electronic Library Services & Applications Lobby (ELSA)

The goals SAIC/ASSET are pursuing involve:

- Creating a focal point for software reuse information exchange
- Advancing the technology of software reuse
- Providing an electronic marketplace for reusable software products to the evolving national software reuse industry.

To achieve these goals, SAIC/ASSET operates the Worldwide Software Reuse Discovery (WSRD) Library. The WSRD Library is populated with quality reusable software components which can be distributed to its subscribers. WSRD contains over 700 assets available to over 1500 users throughout the world. The library specializes in software lifecycle artifacts and documents written specifically to promote software reuse and development. SAIC/ASSET users have access to other components stored in the CARDS and DSRS reuse libraries. Through the WSRD, users can search, browse and download asset catalogs in over 30 domains. SAIC/ASSET's World Wide Web pages, located at http://source.asset.com/, describe products and services offered through SAIC/ASSET, as well as information related to software reuse.

- **The Public Ada Library (PAL)**

Since 1984, the Ada Software Repository (ASR) has been a major, publicly available source of Ada code. Now called the Public Ada Library (PAL) [27], it provides more than 100 megabytes of programs, components, tools, general information, and educational materials on Ada. It also contains materials on the Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL), which is based on Ada.

For those with access to the Internet, the PAL can be accessed via the File Transfer Protocol (FTP). The PAL is located on the wuarchive.wustl.edu host, and on mirror sites at ftp.cnam.fr and ftp.cdrom.com. Also, the PAL can be obtained on disk, tape, and compact-disk read-only memory (CD-ROM).

Additionally, the PAL can be accessed by means of such Internet services as: the Network File System (NFS), which allows computers to share files across a network; archive, a system of querying anonymous-FTP sites; and gopher, via gopher servers wuarchive.wustl.edu and gopher.wustl.edu.

- **CAPS Software Reusable Component Repository**

CAPS (Computer Aided Prototyping System) is a research project developed by the Software Engineering Group led by Prof. Luqi at Naval Postgraduate School [39]. Initial implementation of CAPS software base was first explored in 1988 [40]. An implementation of the software base was accomplished in 1991 by using ONTOS, an object oriented data base management system that provides an interface to C++ for customization and flexibility [41]. The CAPS software base is being changed to a software component repository since 1998 [1]. The CAPS component repository supports two critical functions, component storage and component retrieval. Much effort has been made to improve the component retrieval method [42, 43]. To the best of our knowledge, CAPS Repository is the only one that supports profile matching and signature matching. It provides high precision and recall retrieval method at same time [1]. The CAPS repository is still under construction. A prototype has been developed to verify the performance of the retrieval methods [1].

- **The Ada Library and the Reuse Library at the Defense Information Systems Agency (DISA)**

The Ada Library and the Reuse Library at the Defense Information Systems Agency (DISA) are public, non-lending, reference libraries for all professionals, students, and researchers seeking information on the Ada programming language and on software reuse [37]. The number of books and articles on Ada and on reuse grows daily. Also, there is a wealth of information available on the Internet and on the World Wide Web. Putting the Net together with the Ada and Reuse Libraries makes a very powerful research tool.

Both Libraries collect and hold information found in documents, books, conference proceedings, newspaper and journal articles, and other multimedia material.

The Libraries can provide assistance in two ways: helping users find publications in each library, and conducting on-line searches for published information available elsewhere. Users can access these resources in person, and via the Web, or they can call DISA to request a search.

Over the Web, go to http://sw-eng.falls-church.va.us. There, click on "Library" at the main page of either the AdaIC or the ReuseIC. Users can search database by title, author, subject, or publisher.

5. Comparison

Commercial reusable component repositories usually are integrated into a CASE environment [28, 29]. Currently, some major repositories (ASSET, PAL, and DSRS) begin to use web-based techniques to provide services. They are utilizing flat files written in HyperText Markup Language (HTML). Electronic Library Services

and Applications (ELSA) has gone a step further by using the Multimedia Oriented Repository Environment (MORE).

Following are some comparison results for all the repositories listed above.

Features	Web-Based	Integrated into CASE Environment	Security Control	Retrieval Methods
+1 Reuse Repository		Y	Y	Browsing
SALMS	Y	Y	Y	Keywords
ASRR			Y	Keywords
The Universal Repository		Y	Y	Browsing and Keywords
AIRS			Y	Facets Approach
RLT			Y	Keywords
HSTX Reuse Repository	Y		Y	Keywords
DSRS	Y		Y	Keywords
LID			Y	Keywords
I-CASE		Y	Y	Keywords
MORE	Y		Y	Keywords
ASSET	Y	Y	Y	Keywords
PAL	Y		Y	Keywords
CAPS		Y		Browsing, Keywords, Profile & Signature Matching
Ada Library and Reuse Library (DISA)	Y		Y	Browsing and Keywords

6. Conclusion

Web-based reuse is the trend of software component repositories supported by the government. To be a part of an integrated CASE environment is the trend of commercial software component repositories. Usually, the aim of the first one is to provide a service within a domain, organization, or area, such as ASSET for DoD, DSRS for DISA etc. This kind of repository is used in a wide scope. The aim of the second is to provide an integrated CASE environment for a software development organization. So, this kind of repository is generally a part of CASE environment and is used in a relatively narrow scope.

The long-term goal of the CAPS project [1] is to provide a distributed software component repository to support the development of prototype systems through intranet technology. So, it will combine the advantages of commercial component repositories and government supported repositories. This developing research system is an example of future software repositories.

References

[1] Luqi and Jiang Guo, "Toward Automated Retrieval for a Software Component Repository", Proceedings of IEEE International Conference and Workshop on the Engineering of Computer Based Systems (IEEE ECBS), Nashville, USA, March 7-12, 1999. Pp. 99-105.

[2] A. Mili, R. Mili, and R.Mittermeir, ``Storing and retrieving software components: A refinement based system," in Proc. 16th Int'l Conf. on Software Engineering, (Sorrento, Italy), pp. 91-100, May 1994.

[3] B. Fischer, M. Kievernagel, and W. Struckmann, ``VCR: A VDM-based software component retrieval tool," in Proc. ICSE-17 Workshop on Formal Methods Application in Software Engineering Practice, 1995.

[4] J. Penix, P. Baraona, and P. Alexander, ``Classification and retrieval of reusable components using semantic features," in Proceedings of the 10th Knowledge-Based Software Engineering Conference, pp. 131-138, Nov. 1995.

[5] A. M. Zaremski, Signature and Specification Matching. PhD thesis, Carnegie Mellon University, Jan. 1996.

[6] A. M. Zaremski and J. M. Wing, ``Specification matching of software components," in 3rd ACM SIGSOFT Symposium on the Foundations of Software Engineering, Oct. 1995.

[7] J. Penix and P. Alexander, ``Design representation for automating software component reuse," in Proceedings of the first international workshop on Knowledge-Based systems for the (re)Use of Program libraries, Nov. 1995.

[8] R. McDowell, and J. Solderitsch, ``The Reusability Library Framework," Proceedings of the Unisys Defense Systems Software Engineering Symposium, January 1990.

[9] R. McDowell and K. Cassell, ``The RLF Librarian: A Reusability Librarian Based on Cooperating Knowledge-Based Systems," Proceedings of the 4th Annual Rome Air Development Center Knowledge-Based Software Assistant Conference, Utica, NY, September 1989.

[10] Eichmann, D., T. McGregor and D. Danley, "Integrating Structured Databases Into the Web: The MORE System," First International Conference on the World Wide Web, Geneva, Switzerland, May 25-27, 1994, pages 369-378.

[11] M. A. Durnin, K. Terry, and R. Sullins, ``Establishing a Repository for Enterprise Wide Software Reuse," in Proceedings Fifth Annual Workshop on Software Reuse Education and Training, July 29 - August 1 1996.

[12] G. Arango and R. Prieto-Diaz, "Domain Analysis Concepts and Research Directions", Domain Analysis and Software System Modeling, R. Prieto-Diaz and G. Arango eds., IEEE Computer Society, 1991.

[13] J.-J. Jeng and B. H. C. Cheng, ``A formal approach to using more general components," in Proceedings of the 9th Knowledge-Based Software Engineering Conference, pp. 90-97, September 1994.

[14] DSRS - Defense Technology for Adaptable, Reliable Systems URL:http://ssed1.ims.disa.mil/srp/dsrspage.html

[15] STARS - Software Technology for Adaptable, Reliable Systems URL: http://www.stars.ballston.paramax.com/ index.html

[16] D. E. Perry and S. S. Popovitch, “Inquire: Predicate-based use and reuse," in Proceedings of the 8th Knowledge-Based Software Engineering Conference, pp. 144-151, September 1993.

[17] D. Garlan, "Research Directions in Software Architecture", ACM Computing Surveys, 27(2), June 1995.

[18] R. Girardi, "Towards Effective Software Abstractions for Application Engineering", in Procs. NASA Focus on Reuse workshop, Sept. 1996.

[19] Asset Library Open Architecture Framework, Version 1.2; STARS-TC-04041/001/02; 14 August 1992.

[20] ELSA - Electronic Library Services & Applications URL: http://rbse.mountain.net/ELSA/

[21] L. S. Levy, "A metaprogramming method and its economic justification", IEEE Trans. Softw. Eng. SE-12(2), Feb. 1996, pp. 272-277.

[22] R. Girardi and B. Ibrahim, "Using English to Retrieve Software", The Journal of Systems and Software, Special Issue on Software Reusability September 1995.

[23] R. Girardi, "Classification and Retrieval of Software through their Descriptions in Natural Language", Technical report, University of Geneva - CUI, December 1995.

[24] R. T. Price and R. Girardi. A class retrieval tool for an object-oriented environment. In Procs. 3rd Conf. Technology on object-Oriented Languages and Systems, pages 26-36, November 1990.

[25] M. Simos, “The Growing of an Organon: A Hybrid Knowledge-Based Technology and Methodology for Software Reuse," Proceedings of 1988 National Institute for Software Quality and Productivity (NISQP) Conference on Software Reusability, April 1988, pp. E-1 through E-25.

[26] ASSET - Asset Source for Software Engineering Technology URL: http://source.asset.com/asset.html

[27] PAL - Public Ada Library URL: http://web.cnam.fr/Languages/Ada/PAL/

[28] CARDS - Comprehensive Approach to Reusable Defense Software URL: http://dealer.cards.com/

[29] COSMIC - NASA's Software Technology Transfer Center URL: http://www.cosmic.uga.edu/

[30] J.-J. Jeng and B. H. C. Cheng, ``Using formal methods to construct a software library," in Proceedings of 4th European Software Engineering Conference, Lecture Notes in Computer Science, vol. 717, pp. 397-417, September 1993.

[31] +1 Software Engineeering Corporate Mission, URL http://www.plus-one.com/company .html

[32] Elisabetta Morandin, “SALMS v5.1: A System for Classifying, Describing, and Querying about Reusable Software Assets”, The Proceedings of 5th International Conference on Software Reuse (ICSR '98).

[33] Project Management Tool Suite System (Automated Software Reuse Repository), URL http://wv.ewa.com-/srr_overview.html

[34] Universal Repository, URL http://www.marketplace.unisys.com/urep/

[35] Reuse Library Toolset of EVB Software Engineering, http://gopher.metronet.com:70/0/newprod/by-vendor/E-/evb_software_e/941208.01

[36] The HSTX Software Reuse Repository, selsvr.stx.com/~eryq/swreuse/home.html

[37] STARS Q9 BASELINE Ada LIBRARY, Technical Reports on the Software Reuse CFCSE-IC http://dii-sw.ncr.disa.mil/ReuseIC/guidelines/ReusabilityGuidelines.html

[38] Robert Rutherford, “Reuse on I-CASE,” Proceedings of Fifth Annual Workshop on Software Reuse Education and Training, July 29, 1996 - August 1, 1996. http://www.asset.com/WSRD/conferences/proceedings/summary/-summary.html

[39] Luqi, and M. Ketabchi, “A Computer-Aided Prototyping System,” IEEE Transactions on Software Engineering, October 1988.

[40] Robert Steigerwald, Luqi, and John McDowell, “CASE Tool for Reusable Software Component Storage and Retrieval in Rapid Prototyping,” Information and Software Technology, pp. 698-705, 1991.

[41] Scott Dolgoff, “Automated Interface for Retrieving Reusable Software Components”, Master’s Thesis, Naval Postgraduate School, September 1992.

[42] Doan Nguyen, “An Architectural Model for Software Component Search”, Ph.D. Dissertation, Naval Postgraduate School, December 1995.

[43] Jeffrey Herman, “Improving Syntactic Matching for Multi-Level Filtering,” M.S. Thesis, Naval Postgraduate School, September 1997.

[44] Rubin Prieto-Diaz, “Implementing Faceted Classification for Software Reuse”, Communication of the ACM, pp. 89-97, May 1991.