Software Reuse Libraries with Mosaic

Jeffrey S. Poulin and Keith J. Werkman

Loral Federal Systems–Owego
MD 0210, Owego, NY 13827
Tel: (607) 751-6899/4889, fax: (607) 751-6025
Email: poulinj,keithw@lfs.loral.com

Abstract

This paper describes a Reusable Software Library (RSL) interface and search tool implemented using Mosaic. Mosaic provides a simple, easy-to-use method to find and extract reusable assets from a RSL, allows distributed access to assets from a variety of platforms, and can support most of the features of formal RSLs without any modifications. Through the use of HyperText Markup Language (HTML) forms, we implemented functions normally found in commercial-grade RSLs, such as component search, user registration, and problem reporting. Automatic generation of HTML pages and the use of command scripts further allowed us to provide different views of the RSL, such as search by subject. Finally, integrating the RSL with Wide Area Information Search (WAIS) provided a keyword search with minimal effort. Our Mosaic RSL cost less than 1% of the cost to develop a standard RSL and has quickly gained favor due to its intuitive interface and simple yet powerful information retrieval tools. [1]

Keywords: Software Reuse, Reusable Software Libraries (RSL), World Wide Web (WWW), Mosaic.

[1] Proceedings of the *2nd International World Wide Web Conference: Mosaic and the Web,* Chicago, Illinois, 17-20 October 1994.

1 Background

The quest for ways to improve the software development process has led many organizations to pursue the substantial benefits available through software reuse. To this end, these organizations have given a lot of attention to technologies that facilitate reuse; application generators, domain analysis techniques, formal methods, and application frameworks. Many organizations focus their reuse initiatives on a reuse library where members of the organization can both store reusable assets and retrieve assets when they need them. Traditional RSLs use specialized methods for component classification, search, and retrieval. Unfortunately, these formal tools and techniques require both a large investment to implement and substantial training to use. For these reasons, many organizations have seen little use of their RSLs even though they may contain a large number of quality assets.

This paper describes a software reuse library interface and search ability using Mosaic [3, 4]. We developed this interface for the Loral Federal Systems Group RSL, which we refer to as the *Federal Reuse Repository* (FRR). Mosaic provides an simple, easy-to-use method to search for and extract reusable assets from the FRR. With large organizations investing as much as 80 to 130 person-years to develop a formal RSL, the Mosaic interface cost less than 1% of the cost to develop and maintain a standard, commercial-quality RSL [2, 11]. Nonetheless, the Mosaic-based RSL has quickly gained favor due to its intuitive interface and powerful yet simple features.

The Mosaic interface allows us to take advantage of the existing AIX^R (UNIX^R) file structure and to generate multiple views of the RSL, thereby allowing users to search for assets in several ways. First, we provide a hierarchical view based on the original source of the asset. Second, we provide an "asset by subject" view of the same information. Third, we have implemented a keyword search using WAIS. Finally, we have found that the tools and features of Mosaic such as forms and the Common Gateway Interface (CGI) allow us to implement a variety of common RSL functions such as registering users of RSL components, logging statistics of module usage, and automatically notifying users of the RSL (via e-mail) of component updates, problem reports, and other RSL-related information.

2 Reusable Software Libraries

The original Loral Federal Systems RSL provided a central repository for sharing, managing, and reusing software-related products across Loral Federal Systems sites. A copy of the RSL ran at each site and operated in cooperative fashion with the RSLs at the other sites. Together, the RSLs established a system of shared libraries; any organization could establish a library to service the needs of a department, project, business area, or higher organization.

The RSL ran on IBM's two major mainframe operating systems, Multiple Virtual Storage (MVS)^R and Virtual Machine (VM)^R, because nearly every member of the company has access to these systems. Although the RSL interface complied with the IBM Systems Application Architecture (SAA)^{TM} and Common User Access (CUA)^{TM} interface standards, developers repeatedly expressed the desire for a workstation-style Graphical User Interface (GUI). This need increased in priority as development work moved almost exclusively to workstation platforms.

The search mechanism implemented in the RSL used a detailed classification scheme based on the work of Prieto-Diaz and Freeman [12, 13]. To locate a component for reuse, a user invoked the RSL tool and specified "facets," or pre-defined software attributes, and acceptable values for those facets. The tool executed a search of the RSL database for components classified with facet values equal to those specified by the user. If the user felt satisfied with any of the components located by the search, the user copied the component from the RSL onto the user's local disk space. A detailed discussion of the classification scheme and the issues surrounding this kind of classification appears in [10].

In theory, this extensive and formal mechanism provides detailed information upon which a user can search for and assess the usefulness of reusable components. However, quite often the quantity and formality of the information only serves to confuse the user. First, the up-front presentation of large amounts of classifiers makes it difficult to quickly extract the key bits of needed information. Second, having a formal classification scheme requires users to receive training in its use; untrained users will not effectively use the mechanisms so carefully provided to assist them [14]. Users of the RSL demanded a simple, easy to use keyword search that although might lack the precision of a formal mechanism, required little or no training to use.

3 Requirements for the RSL

Users of the existing RSL made the first two requirements for our replacement implementation explicit:

1. It had to have a friendly, GUI interface.
2. Its use had to come naturally and intuitively.

The remaining major requirements consisted of a set of constraints and desired features.

3.1 Constraints

Like many companies facing highly competitive markets, we sought the most efficient and cost-effective means to manage our information. The implied task involved porting the contents of the RSL tool from the mainframe to a workstation-based environment. Furthermore, we needed a way to not only locate reusable software but also a way to locate other items of interest such as key personnel, information about ongoing programs, the latest developments in various technologies, and trade studies previously conducted by our company. The desired information retrieval tool had to handle all these types of information.

Since many Loral Federal Systems customers require compliance with "Open Systems" platforms, much of our software development staff works in an AIX or UNIX environment. However, we also have a large population who work with and integrate local area networks based on the DOS or OS/2^{TM} operating systems. Finally, we required compatibility with the host legacy systems, especially for support personnel. We sought an information retrieval tool accessible by users on all these platforms and one that did not restrict the user's access to information which happens to reside on any other platform.

Of course, we especially desired a system that would cost very little to build and maintain.

3.2 Desired Features

The detailed classification scheme described above brought with it the burdensome process of manually classifying every component a user entered into the RSL. To avoid this, we wanted the ability to automatically or semi-automatically index components in the RSL, even if this meant losing a detailed search ability for the RSL. However, users felt that keyword searches adequately met their search needs despite their poor precision relative to that available with facets. With all the power resting on their desktops, users felt they could afford several attempts at locating a suitable component.

Because the tool would provide a search mechanism for more than just code the tool needed to have the ability to launch other tools such as browsers, viewers, and programs that handle diverse kinds of information such as multi-media. From security reasons we also needed an access control ability to limit access to the RSL to authorized users. Finally, we considered several additional features that we wanted to have:

- *Problem reporting for reusable assets.* The ability for a user to send e-mail to the owner of a component or the librarian should the user require service on a component extracted from the RSL.
- *Registration of users of assets.* The ability for a library administrator to locate users of assets for the purpose of recording use of the RSL, to report possible problems with extracted components reported by other reusers, and to aid in configuration management of extracted components as newer versions become available.
- *Version control of assets.* The ability to manage several versions of the same assets.

Finally, everyone insisted that the tool have very good performance.

4 Possible Solutions

We reviewed a number of specialized reuse library tools against the above requirements and now discuss some of the results. Among the tools we considered, the *ReDiscovery*^{TM} information retrieval tool from IBM allows users to create and search meta-databases of information about virtually any kind of database

or file system. However, ReDiscovery requires the user to create and maintain these meta-databases and it only runs on OS/2.

We briefly looked at two IBM internal use tools but dropped them for consideration when IBM sold their Federal Systems Division (to which we belong) to the Loral Corporation. We first considered an OS/2 LAN-based tool, called the Reuse Library System (RLS), which provides users with a easy-to-use GUI interface components stored on an OS/2 LAN. The second of these tools, called *XGuru,* automatically indexes any collection of text files using a highly intelligent information retrieval algorithm [7]. Created by IBM research and in use at several IBM locations, XGuru allows the user to ask natural language queries. By using the same algorithm used to index the information XGuru returns candidate files in a ranked order of preference.

The public domain contains a specialized tool called the STARS Reuse Library (SRL), one product of the United States Advanced Research Projects Agency (ARPA) Software Technology for Adaptable, Reliable Systems (STARS) program. The Asset Source for Software Engineering Technology (ASSET) program uses the SRL to manage its reuse library. The SRL has a friendly system of menus and built-in security, browsers, and librarian tools. However, it requires an underlying Oracle^{TM} database and has no GUI interface [1].

Finally, we considered the *InQuisiX*^{TM} reuse library from the Software Productivity Solutions Corporation. SPS sells the InQuisix tool to companies who want a full function library tool. It met our requirements for GUI interface, distributed data management, and integration of multiple media types, but only ran on a limited set of platforms. The greatest disadvantage of InQuisix came from its requiring a license for every user of the tool, thereby leading to a cost we did not want to incur [5].

Among the options for free, shareware, or public-domain tools, we considered numerous utilities that have recently become popular for resource discovery on the Internet [8]. As opposed to specialized library tools and commercial products each of these tools runs on a wide range of operating systems and platforms. The popularity of these products, as demonstrated by the their explosive growth and wide distribution, gave us confidence in their use despite the lack of official 'support.'

Anonymous FTP sites provided perhaps the simplest and cheapest overall solution. A common method to distribute software and one that requires no special tools, it lacks a user-friendly interface and requires some training to use in its basic form. FTP also has no search ability and works best only when the user knows exactly where and for what to look.

Archie and Gopher both provide client-server methods of traversing pre-established lists of information at distributed sites. Both use primarily character interfaces, even in their GUI versions. Although Gopher does not have a search ability, Archie searches files at anonymous FTP sites for a user-specified filename or part of a filename; it does not search file contents. We would have to use additional tools, such as VERONICA, to provide text search ability.

Finally, the tool that met nearly all of our requirements came in the form of Mosaic. Mosaic provides a client-server method of traversing pre-established menus (presented to the user as "pages") via hypertext links. Although Mosaic does not have a search ability, WAIS provides a keyword search that integrates tightly with the Mosaic interface. Mosaic runs on every platform we required, handles multi-media, has the ability to launch external applications and viewers. The HTML forms capability allows us to easily implement most of the reuse related features (such as user registration and problem reporting) that do not come by default with the stock Mosaic server (such as usage statistics).

5 Approach using Mosaic

Porting the contents of the RSL from the mainframe to distributed platforms demanded reorganization of the FRR assets from the flat file system on the mainframe to the hierarchical system used by AIX systems. Each FRR component normally comes in its own file, along with up to 15 files containing information supporting its use; e.g., design specifications, an abstract, and integration instructions. We made the natural choice to store an asset and its supporting information together in one AIX directory. We then chose to organize the groups of assets based on source; e.g., a parent directory for all components supplied by Program *A* or licensed from Company *B*. Although this organization may not necessarily make it easy for reusers to locate assets, it made control of the assets much easier. We decided to provide alternate views (or indices) of the FRR, such as grouping assets by function, using html pages. The use of the hierarchical file system not only allowed a very nice way to organize the FRR contents, but it obviated the need for an underlying database

system to manage the data and therefore required little investment and effort.

The second major consideration concerned security. Again we relied on the safeguards already in place in our environment; Andrew File System (AFS) Access Control Lists (ACLs) and standard AIX file permissions allowed us to grant and deny access to the FRR. We actually authorize users through the use of subnet masks based on Internet Protocol (IP) address; this allows us acceptable level of access control to users throughout Loral Federal Systems.

Loral FS Reuse

You've reached the WWW page for Loral Federal Systems Reuse, home of the Federal Reuse Repository (FRR). The FRR contains several hundred reusable software and document components; click here for information about the FRR.

 FRR, hierarchical view (arranged by Language/Library)

 FRR, arranged by Subject

 Search the FRR for needed components (via WAIS)

Click here to find out more about the LFS Reuse group or to send them mail.

Figure 1: The FRR Home Page

The figures show the implementation of the FRR using Mosaic. Figure 1 shows the FRR Home Page. As shown, we currently provide three ways to browse the FRR; (1) by a hierarchical view which mirrors the organization in the AIX file system, (2) by subject based on the major function or service provided by the component, and (3) via keyword search using WAIS.

We sorted the hierarchical view based first on implementation language. If the user selects the Ada programming language, the mosaic page shown in Figure 2 appears. This page lists the various sources of reusable Ada software. Selecting "Circuit Card Assembly and Processing System (CCAPS)," one of the programs that produced software for the FRR, results in the page shown in Figure 3.

Note that *monitors* and a *semaphore* both appear as components supplied from the CCAPS program. This hierarchical approach shows one way a user can get to this information. The user can also get to these same components using the second method on the FRR Home Page; Figure 4 shows the page produced if the user elects to search the *FRR, arranged by Subject.* Note that *Synchronization Components* appears on the FRR component listing by subject. Since monitors and semaphores both provide synchronization functions, the user will find them on the *Synchronization Components* page along with other monitors, events, barriers,

Ada Reuse Collections

Select the library you need:

- Advanced Automation System (AAS)
- Ada Run Time Environment (ARTE)
- Booch − Ada
- Booch − Ada Enhanced
- Common Ada Missile Packages (CAMP)
- Circuit Card Assembly and Processing System (CCAPS)
- General Purpose Ada
- Karlsruhe
- Realtime And Distributed Ada Services (RADAS)

Figure 2: The Ada Language Home Page

locks, pulses, and related Ada language synchronization from sources other than CCAPS.

The third search option consists of a standard ISINDEX WAIS search form. Because WAIS integrates well with xmosaic, we selected it for simple keyword searches of the FRR. By using the classification data from the original RSL to index the WAIS database we feel we have achieved a reliable index for keyword searches.

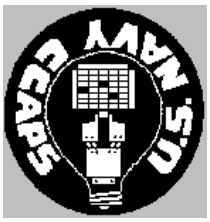
6 Related Work

The Repository Based Repository for Software Engineering (RBSE) research and development group has developed the Multimedia Oriented Repository Environment (MORE) using Mosaic and the Web as its sole user interface. MORE provides client browsing, search, repository definition, and data entry through Web clients. Unlike the system in this paper, MORE uses a meta-data based repository, or a database that contains information about the reusable assets rather than the actual assets. With the exception of the system home page the MORE dynamically generates the entire user interface [6].

7 Future Work

Our future activities include providing several searching capabilities for retrieval of software modules and associated documentation based on the *Structured Abstract* concept proposed in [9]. The Structured Abstracts will allow users to provide a more detailed search than the simple keywords offered by WAIS by using the classification information previously developed for the original RSL. Using html forms for the Structured Abstracts will also allow us to keep the simple, easy-to-use GUI interface. To provide access to the FRR to users of ascii-only terminals such as 3270 protocol sessions, we plan to implement a Lynx interface.

We will complete implementation of html forms for the "check out/check-in" capabilities of the Software Reuse Library via a back-end database and the webserver CGI. Keeping in mind the desire for a simple, intuitive tool, we plan html forms for component user registration, problem reporting, and submission forms



CCAPS

Circuit Card Assembly and Processing System

Please review the legal information before using any of these components. If you would like to preview descriptions of the CCAPS components, here is a collection of the CCAPS abstracts.

Select the component you need:

- ada interface to ibm gddm
- generic timer facility
- convert integer to string
- line parse manager
- msgm asm interface
- monitors
- rscs io interface
- semaphore
- system commands

Figure 3: The CCAPS Home Page

for new software modules. In addition, we will also look at integrating documentation (authoring) tools into the RSL environment to allow users to make their own html hyper-linked documentation point to the relevant reusable software modules in the FRR.

8 Acknowledgements

We would like to acknowledge Jim McKinstry for his early work on this project and Allen Matheson of the Cimarron Corporation for his continued support of the FRR and the key role he has had in implementing the work described in this paper. We would also like to thank Gary Kennedy and Tom Loggia of Loral Federal Systems-Bethesda for supporting this project.

9 Cited References

1. Asset Source for Software Engineering Technology (ASSET), Building 2600, Suite 2, 2611 Cranberry Square, Morgantown, WV, 26505, e-mail: *info@source.asset.com.*

2. Berg, Klaus, "CLASSLIB - Class Management and Reuse Support on a MVS Mainframe," *Reusability Track of the 1994 ACM Symposium on Applied Computing (SAC'94),* Phoenix, Arizona, 6-8 March 1994, pp. 53-58.

FRR Component Listing by Subject

Select the type of component you need:

- Bit/String Manipulation Components
- Command Line Components
- Communication Components
- Data Structures Components (long list!)
- File Services Components
- Graphics Components
- Input/Output Components
- Miscellaneous Utilities
- Numerics and Math Packages
- OS (POSIX) Interfaces
- Real−Time Components
- Sorting and Searching Routines
- Synchronization Components

Figure 4: The FRR, Arranged by Subject

3. Berners-Lee, Tim, Robert Cailliau, Jean-Francois Groff, and Bernd Pollermann, "World-Wide Web: The Information Universe," *Electronic Networking: Research, Applications and Policy,* Vol. 11, No. 2, Meckler, Westport, CT, Spring 1992.

4. Berners-Lee, Tim, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielen, and Arthur Secret, "The World-Wide Web," *Communciations of the ACM,* Vol. 37, No. 8, August 1994, pp. 76-82.

5. Donaldson, Cameron, "InQuisiX: An Electronic Catalog for Software Reuse," *SIGIR Forum,* Vol. 28, No. 1, Spring 1994, pp. 8-12.

6. Eichmann, David, Terry McGregor, and Dann Danley, "Integrating Structured Databases Into the Web: The MORE System," *First International Conference on the World Wide Web,* Geneva, Switzerland, 25-29 May, 1994.

7. Maarek, Yoelle S., Daniel M. Berry, and Gail E. Kaiser, "An Information Retrieval Approach for Automatically Constructing Software Libraries," *IEEE Transactions on Software Engineering,* Vol. 17, No. 8, August 1991, pp. 800-813.

8. Obraczka, Katia, Peter B. Danzig and Shih-Hao Li, "Internet Discovery Services," *IEEE Computer,* September 1993, pp. 8-22.

9. "Automatic Generation of a Rapid Assessment Aid for Reusable Components," Jeffrey S. Poulin *Research Proposal*, Loral Federal Systems-Owego, Owego, NY, 18 July 1993.

10. Poulin, Jeffrey S. and Kathryn P. Yglesias, "Experiences with a Faceted Classification Scheme in a Large Reusable Software Library (RSL)," *Seventeenth Annual International Computer Software and Applications Conference,* Phoenix, AZ, 3-5 November 1993, pp. 90-99.

11. Poulin, Jeffrey S., "Populating Software Repositories: Incentives and Domain-Specific Software," to appear, *Journal of Systems and Software,* Spring 1995.

12. Prieto-Diaz, Ruben and Peter Freeman, "Classifying Software for Reusability," *IEEE Software,* Vol. 4, No. 1, January 1987, pp. 6-16.

13. Prieto-Diaz, Ruben, "Implementing Faceted Classification for Software Reuse," *Communications of the ACM,* Vol. 34, No. 5, May 1991, pp. 88-97.

14. Yglesias, Kathryn P., "Limitations of Certification Standards in Achieving Successful Parts Retrieval," *Proceedings of the 5th International Workshop on Software Reuse,* Palo Alto, California, 26-29 October 1992.

10 Vitas

Jeffrey S. Poulin, Ph.D.

Dr. Poulin works with the Advanced Technology department of Loral Federal Systems-Owego where he serves as Principal Investigator (PI) for Open Systems Environment Independent Research and Development (IRAD). His past Loral Federal Systems-Owego responsibilities include:

- Reuse Strategy, Sustaining Base Information Services (SBIS) program.
- SBIS Engineering and Technology Team Lead, Integrated Software Engineering Development Environment, Open Systems Development group.

Dr. Poulin formally served with the IBM corporate Reuse Technology Support Center (RTSC) where his responsibilities included reuse standards, economics, and legal issues. As part of his reuse metrics work, he helped lead the development of the IBM reuse measurements and return on investment (ROI) model. His background includes semantic data modeling in object-oriented database systems with a focus on support for Computer Aided Software Engineering (CASE). He participates in the Association for Computing Machinery and the IEEE Computer Society. A Hertz Foundation Fellow, Dr. Poulin earned his Bachelors degree at the United States Military Academy at West Point and his Masters and Ph.D. degrees at Rensselaer Polytechnic Institute in Troy, New York.

Keith J. Werkman, Ph.D.

Dr. Werkman, formerly employed by IBM's Federal Systems Company in Owego, NY, now participates as a member of Loral Federal Systems-Owego's Advanced Technology department assisting as one of the sites resources in artificial intelligence (AI). One of his research tasks integrates AI into a variety of new and existing business areas. Dr. Werkman has worked on developing AI tools with multimedia interfaces to support these enhanced user environments at LFS-Owego. His research interests include using distributed AI to support a variety of user information environments including those for software reuse, agile manufacturing, concurrent engineering, group decision support systems and groupware. He participates in the American Association of Artificial Intelligence (AAAI), Association of Computing Machinery (ACM), ACM SIGART, and IEEE Computer Society.

Dr. Werkman earned his Bachelors and Masters degrees from Lehigh University in Bethlehem, PA. While employed as a National Science Foundation Engineering Research Center (NSF-ERC) Fellow at the ATLSS Center and Lehigh, Dr. Werkman earned a doctorate in Computer Science based on his distributed AI negotiations research. Dr. Werkman has over 30 publications in the areas of Distributed Artificial Intelligence (DAI), Concurrent Engineering (CE), AI in Design, Computer Supported Cooperative Work (CSCW), Design for Manufacturability (DFM) and Enterprise Integration (EI).

For more information, contact:

poulinj@lfs.loral.com or keithw@lfs.loral.com