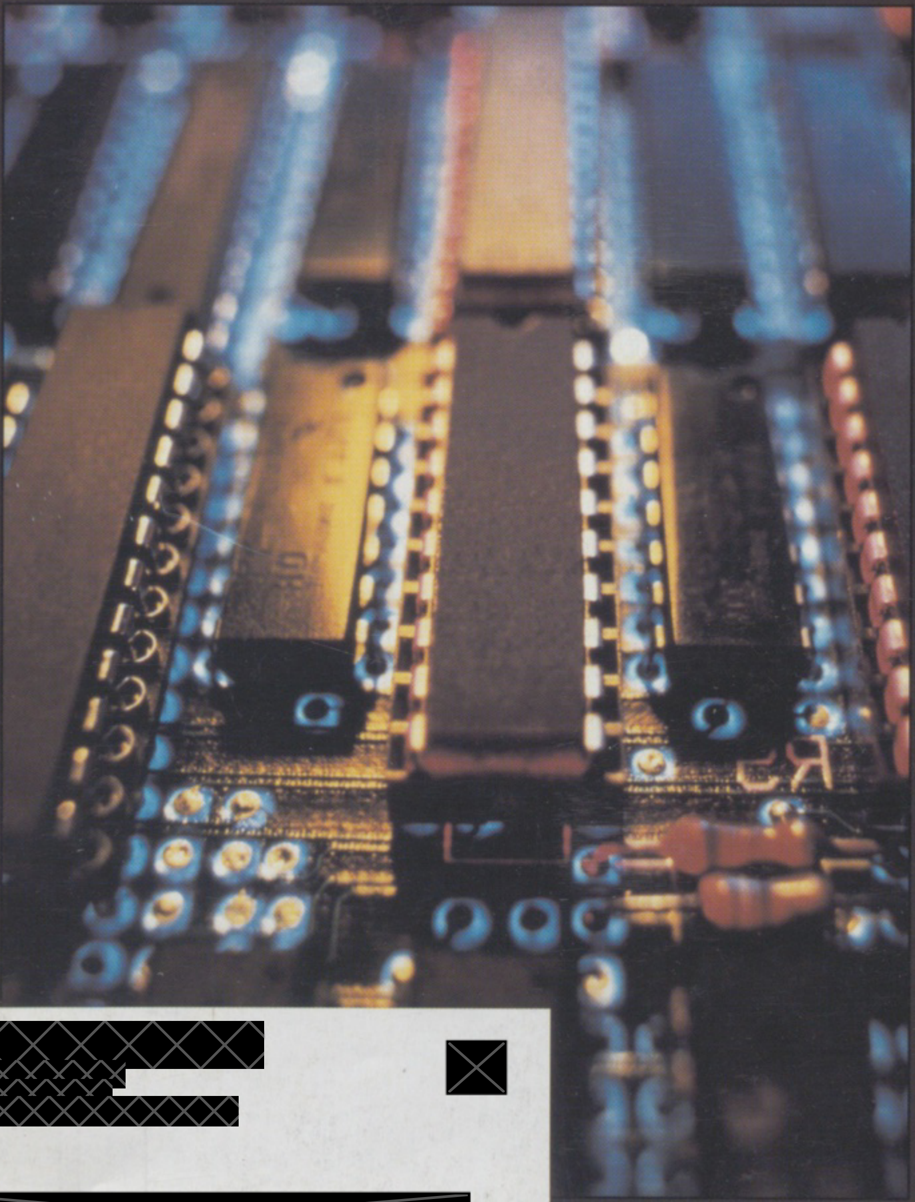
Information Technology AND Libraries

ISSN 0730-9295

Volume 21 Number 1

MARCH 2002



[redacted]

[redacted]

[redacted]

IN THIS ISSUE

- Open Source Software
- Software Reviews

Local control global reach

The OCLC FirstSearch service delivers a world of information to your users, with a focus on your collection. Through unparalleled resources like WorldCat that are enhanced by searching tools and system features you can customize, FirstSearch makes finding information convenient for your users and affordable for your budget. Your users have access to the holdings of thousands of libraries worldwide, full text from more than 9,000 serials (including 3,400 electronic journals), and best of all, the riches of your library's collection.

www.oclc.org

WorldCat
The OCLC Online Union Catalog

OCLC Online Computer Library Center

OCLC—Furthering Access to the World's Information

Dan Marmion, Editor
University of Notre Dame, Notre Dame, Indiana (dmarmion@nd.edu)
Judith Carter, Managing Editor
OCLC, Inc., Dublin, Ohio (carterj@oclc.org)
Tom Zillner, Book Review Editor
Wisconsin Library Services, Madison, Wisconsin (tzillner@wils.wisc.edu)
Andy Boze, Software Review Editor
University of Notre Dame, Notre Dame, Indiana (Boze.1@nd.edu)

Information Technology and Libraries (ISSN 0730-9295) is published quarterly in March, June, September, and December by the American Library Association, 50 E. Huron St., Chicago, IL 60611. It is the official publication of the Library and Information Technology Association, a division of the American Library Association. SUBSCRIPTIONS: Subscription price, $25, is included in membership dues. Nonmembers may subscribe for $50 per year in the U.S.; $55 in Canada, Mexico, Spain, and other PUAS countries; $60 in other foreign countries. Send subscription requests to *Information Technology and Libraries,* Subscription Department, American Library Association, 50 E. Huron St., Chicago, IL 60611; 800-545-2433; fax: (312) 944-2641; subscription@ala.org. Single copies, $15. Periodical-class postage paid at Chicago, Illinois, and at additional mailing offices. POSTMASTER: Send address changes to *Information Technology and Libraries,* 50 E. Huron St., Chicago, IL 60611.

Information Technology and Libraries publishes material related to all aspects of libraries and information technology, including digital libraries, metadata, authorization and authentication, electronic journals and electronic publishing, telecommunications, distributed systems and networks, computer security and intellectual property rights, technical standards, geographic information systems, desktop applications, online catalogs and bibliographic systems, optical information systems, software engineering, universal access to technology, futuristic forecasting, library consortia, vendor relations, and technology and the arts. *ITAL* welcomes unsolicited manuscripts. Submissions should follow the guidelines stated under "Instructions to Authors" in the June 2001 issue or on the LITA Web site at www.lita.org/ital/infoauth.htm. Manuscripts of feature articles, communications, and tutorials should be addressed to Dan Marmion, 221 Hesburgh Library, The University of Notre Dame, Notre Dame, IN 46556-5629. Books for review should be addressed to Tom Zillner, WILS, 464 Memorial Library, 728 State St., Madison, WI 53706. Software for review should be addressed to Andy Boze, Hesburgh Library, The University of Notre Dame, Notre Dame, IN 46556-5629.

ADVERTISING: The Goldman Group, 14497 N. Dale Mabry Hwy., Ste. 205N, Tampa, FL 33618; telephone (813) 264-2772; fax (813) 264-2343.

As a matter of policy, *ITAL,* as the scholarly organ of LITA, does not review LITA publications.

PRODUCTION: ALA Production Services (Troy D. Linker, Kevin Heubusch; Ellie Barta-Moran, Angela Hanshaw, Kristen McKulski, and Karen Sheets), American Library Association, 50 E. Huron St., Chicago, IL 60611.

Abstracted in *Computer & Information Systems, Computing Reviews, Information Science Abstracts, Library & Information Science Abstracts, Referativnyi Zhurnal, Nauchnaya i Tekhnicheskaya Informatsiya, Otdyelnyi Vypusk,* and *Science Abstracts Publications.* Indexed in *CompuMath Citation Index, Computer Contents, Computer Literature Index, Current Contents/Health Services Administration, Current Contents/Social Behavioral Sciences, Current Index to Journals in Education, Education, Library Literature, Magazine Index, NewSearch,* and *Social Sciences Citation Index.* Microfilm copies available to subscribers from University Microfilms, Ann Arbor, Michigan.

The paper used in this publication meets the minimum requirements of American National Standard for Information Sciences—Permanence of Paper for Printed Library Materials, ANSI Z39.48-1992.∞

Information Technology AND Libraries

Volume 21, Number 1 March 2002 ISSN 0730-9295

Abstracts and contents pages of recent issues of *ITAL* can be found on the LITA Web site at www.lita.org/ital/index.htm.

SPECIAL ISSUE: Open Source Software
JEREMY FRUMKIN, Guest Editor

cover design Kevin Heubusch ● **interior design** Dianne M. Rooney

Guest Editorial: Balancing the Playing Field

Jeremy Frumkin

As a member of the University of Arizona's Digital Library Initiative, I see exciting and new developments in library technology, services, and software. Virtual reference services, library portals, electronic reserves, e-books, the Semantic Web, you name it; the list of new methods by libraries and librarians for serving information to the public continues to grow and emerge. These new opportunities also push us to constantly reexamine our core services and products, as well as our underlying philosophies in how and what we serve our customers. In reexamining ourselves, we must look at our relationship with technology and software and our role in the software development process. Up until now, this process has been (primarily) the sole domain of commercial vendors. Vendors are an integral part of the technology environment in libraries, especially in software development. However, it is up to libraries and librarians to ensure that they are equal partners on the software development playing field. We no longer can afford to let technology and technology companies dictate to us what we can do and how we can do it—instead libraries need to dictate the functions and features of the software they use, and take an active role in maintaining our technology ecology. To do this, we need to expand the library community's knowledge of and expertise with software development and technology—otherwise, to use a football analogy, we'll always be the visiting team.

Open source software (OSS) is both a philosophy and a method that can be used to gain that homefield advantage. Simply put, OSS allows the user to customize, augment, change, and enhance the software so that it better meets their needs. It accomplishes this by providing the source code of the software in addition to the executable program. By doing so, anyone who owns a copy of the software can become a developer as well. It does not mean that every user must become a developer; in fact, in most cases, this won't happen. What is important is the opportunity given—the opportunity to examine and contribute to the source code, which, at a minimum, allows libraries to better understand their tools and systems. OSS, in essence, empowers libraries through knowledge and understanding—it brings library values to software.

This issue presents six articles about OSS and libraries. David Bretthauer provides a detailed look at the development and history of the OSS movement. Eric Lease Morgan follows up with an article about the relationship between OSS and libraries, and the possibilities that emerge from a more comprehensive adoption of OSS by the library community. With a slightly different take, Marshall Breeding presents his views on the adoption of OSS in the library automation system arena. The authors of the fourth article present us with a set of case studies about MARC.pm, a piece of software that gives libraries greater control over their MARC records and illustrates the versatility and usefulness of open source code in a library environment. Harry Wagner's paper on the Extensible Open RDF toolkit demonstrates how new research and emerging technologies can move forward using the OSS-development model. Finally, Karen Coyle gives some context to OSS in libraries by relating it to the concept of open standards and shows how this important concept actually has a long history in the library community.

I hope you find this issue interesting and useful. As libraries become digital libraries, we need to ensure that librarians don't become "virtual" as a result. By participating more fully in the development and requirements of the software we use, we allow ourselves to compete on a level playing field, and we ensure our continued stewardship of today's information access. This role reinforces our place in providing high quality, useful content to our customers. Active participation in the library community's technology ecology, especially through understanding and involvement in software development, enables us not only to play the game but also to help shape the rules as well.

Jeremy Frumkin (frumkinj@u.library.arizona.edu) is the Metadata Systems Librarian for the University of Arizona Library's Digital Library Initiatives Group, Tucson.

Open Source Software: A History

David Bretthauer

In the thirty years from 1970 to 2000, open source software (OSS) began as an assumption without a name or a clear alternative. It has evolved into a sophisticated movement that has produced some of the most stable and widely used software packages ever produced. This paper traces the evolution of three operating systems: GNU, Berkeley Software Distribution (BSD), and Linux, as well as the communities that have evolved with these systems and some of the commonly used software packages developed using the open source model. It also discusses some of the major figures in OSS, and defines both free and open source software.

Since 1998, the open source software (OSS) movement has become a revolution in software development. However, the revolution in this rapidly changing field can actually trace its roots back at least thirty years. OSS represents a different model of software distribution with which many are familiar. Typically in the PC era, computer software has been sold only as a finished product, otherwise called a precompiled binary, which is installed on a user's computer by copying files to appropriate directories or folders. Moving to a new computer platform (Windows to Macintosh, for example) usually requires the purchase of a new license. If the company goes out of business or discontinues support of a product, users of that product have no recourse. Bug fixes are completely dependent on the organization that sells the software. By contrast, OSS is software that is licensed to guarantee free access to the programming behind the precompiled binary, otherwise called the source code. This allows the user to install the software on a new platform without an additional purchase and to get support (or create a support mechanism) for a product whose creator no longer supports it. Those who are technically inclined can fix bugs themselves rather than waiting for someone else to do so. Generally there is a distribution mechanism, such as anonymous FTP, that allows one to obtain the source code, as well as precompiled binaries in some cases. There are also mechanisms for which one may pay a fee to obtain the software as well, such as on a CD-ROM or DVD, which may also include some technical support. A variety of licenses are used to ensure that the source code will remain available, wherever the code is actually used.

To be clear, there are several things open source is not—it is not shareware, public-domain software, freeware, or software viewers and readers that are made freely available without access to source code. Shareware, whether or not one registers it and pays the registration fee, typically allows no access to the underlying source code. Unlike freeware and public-domain software, OSS is copyrighted and distributed with license terms designed to ensure that the source code will always be available. While a fee may be charged for the software's packaging, distribution, or support, the complete package needed to create files is included, not simply a portion needed to view files created elsewhere.

The philosophy of open source is based on a variety of models which sometimes conflict; indeed it often seems there are as many philosophies and models for developing and managing OSS as there are major products. This article will review the development of several major open source projects and attempt to note philosophies of individual projects' creators and maintainers.

The history of open source is closely tied to the history of the hacker culture, since hackers have largely sustained this movement. The term hacker is used here in the sense of one who is both a skilled professional programmer and a passionate hobbyist wishing to advance computer science, rather than the definition recently used by the popular press of a destructive system cracker. Eric Raymond's essay, "A Brief History of Hackerdom" gives an excellent overview of the development of the hacker culture.[1]

Cultural and Philosophical Expectations

There are cultural norms common to nearly all mature, sustained open source projects. Eric Raymond has discussed and demonstrated them more thoroughly in "Homesteading the Noosphere" than space allows here, but several bear repeating. Among them:

- Despite license terms which allow anyone to revise source code, there is usually one person (or a very small group of volunteers) who maintains control of the software and incorporates patches, bug fixes, and added features contributed by others as new releases. This person is often the original creator of the software package or has volunteered to succeed the creator and received the creator's blessing to carry the project forward.
- Often an open source project will have a developer's discussion list (which could be an electronic mailing list using software such as mailman or a usenet newsgroup), where people who contribute patches, bug fixes, and new features discuss their ideas and issues.

David Bretthauer (dave.bretthauer@uconn.edu) is the Network Services Librarian, University of Connecticut, Storrs.

- There is usually a separate discussion list for users of the software, who often are not very technically oriented and who do not normally contribute to the source code but who can report problems and ask for help, both from other users and any developers or maintainers who monitor that list.
- The project and software have a Web site dedicated to that project. The package site may or may not have its own domain name (for example, www.apache.org).
- Maintainers announce new releases of software at such Web sites as freshmeat (http://freshmeat.net) and sourceforge (http://sourceforge.net).
- While documentation is part of a package, widely accepted packages will often have additional written material in books from trade publishers such as O'Reilly and Sam's.[2]

Richard Stallman, GNU, and the Free Software Foundation

Richard Stallman does not identify himself as part of the OSS movement, preferring instead the term "free software."(An explanation of the differences between OSS and free software are detailed below in the section Development of the Term "Open Source.") Regardless of this distinction, Stallman is responsible for laying much of the groundwork for what has become the open source movement. He worked as a programmer at the Artificial Intelligence Lab (AI Lab) at MIT in the 1970s and early 1980s, using a locally developed operating system called Incompatible Timesharing System (ITS). He describes his work situation:

> I had the good fortune in the 1970s to be part of a community of programmers who shared software. Now, this community could trace its ancestry essentially back to the beginning of computing. In the 1970s, though, it was a bit rare for there to be a community where people shared software. And, in fact, this was sort of an extreme case, because in the lab where I worked, the entire operating system was software developed by the people in our community, and we'd share any of it with anybody. Anybody was welcome to come and take a look, and take away a copy, and do whatever he wanted to do. There were no copyright notices on these programs. Cooperation was our way of life. And we were secure in that way of life. We didn't fight for it. We didn't have to fight for it. We just lived that way. And, as far as we knew, we would just keep on living that way.[3]

When he wanted to improve upon the printer driver for a laser printer with a tendency to jam that Xerox had given MIT, he was unable to because Xerox had not and would not supply a copy of the source code. Furthermore, he could not get a copy of the source code from a colleague at Carnegie Mellon because that colleague had signed a nondisclosure agreement with Xerox. When Stallman was not permitted to improve upon the software, he later said, "This was my first encounter with a nondisclosure agreement, and I was the victim. . . . nondisclosure agreements have victims. They're not innocent. They're not harmless."[4]

Eventually the computer system used by the AI Lab was replaced, making all of their lab's previous coding obsolete and forcing the lab to use a new, proprietary operating system. Stallman watched the collapse of his programmer community and began to look for an alternative. That search led him to the concept of free software. He decided to create an operating system complete with all necessary software tools, such as editors, compilers, and utilities, that should be UNIX-compatible so that programmers could use it without having to learn a new operating system. He settled on GNU (pronounced "guh-NEW") as a name for this operating system, a recursive acronym for "GNU's Not UNIX."

In January 1984, he resigned his position at MIT to begin developing GNU. He resigned so that MIT would not be able to interfere with the distribution of GNU as free software. However, Professor Winston, then head of the MIT AI Lab, invited him to continue to use his former office and facilities, and thus he began to develop the pieces. In early 1985, Stallman released the first piece that other programmers were interested in using, an editor called GNU Emacs. He made it available without charge via anonymous FTP; at that time, however, access to the Internet was not very common. As an alternate means of distributing the software, he offered to send people the package on tape for $150. Within a few months he was receiving eight to ten orders per month, which allowed him to pay his living expenses. "So, that was fine, but people used to ask me, 'What do you mean it's free software if it costs $150?' . . . The reason they asked this was that they were confused by the multiple meanings of the English word free. One meaning refers to price, and another meaning refers to freedom. When I speak of free software, I'm referring to freedom, not price. So think of free speech, not free beer."[5]

Stallman defines free software as possessing four essential freedoms:

- You have the freedom to run the program for any purpose.
- You have the freedom to modify the program to suit your needs. (To make this freedom effective in practice, you must have access to the source code, since making changes in a program without having the source code is exceedingly difficult.)
- You have the freedom to redistribute copies, either gratis or for a fee.

- You have the freedom to distribute modified versions of the program, so that the community can benefit from your improvements.[6]

Beyond the concept of freedom, another interesting development was that as the number of programmers using GNU Emacs increased, some of them started to send him messages noting bugs in his source code and offering fixes. Others sent new source code to add new features. Soon these kinds of messages "were pouring in on me so fast that just making use of all this help I was getting was a big job. Microsoft doesn't have this problem."[7]

Stallman also made a practice of incorporating source code written by others wherever possible. Often the determining factor for including the work of others or not was the terms of distribution. For example, this was behind his decision to use the X Window graphical user interface as part of GNU, rather than writing a new windowing system from scratch.

With work on GNU progressing, Stallman needed a way to protect his work from being taken and used in proprietary packages. To ensure this protection, Stallman developed the general concept of copyleft. Traditionally, software is made available either by its author releasing it into the public domain or by closing the source code and using copyright and licensing terms to protect it so it cannot be modified. Each presented a problem for Stallman: releasing software into the public domain means anyone can take it and appropriate it for their own use, including copyrighting it themselves and licensing it as a proprietary product. Releasing it with restrictive copyright and license terms prevents the entire user-review, bug-fix, and feature-addition mechanism that Stallman had found valuable:

> To copyleft a program, we first state that it is copyrighted; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the program's code or *any program derived from it* but only if the distribution terms are unchanged. Thus the code and the freedoms become legally inseparable. Proprietary software developers use copyright to take away the users' freedom; we use copyright to guarantee their freedom. That's why we reverse the name, changing "copyright" into "copyleft."[8]

The specific method Stallman used to copyleft GNU was a licensing agreement he developed called the GNU General Public License (GNU GPL). The first version was released in 1989; the second and current version was released in 1991.

To support the development of GNU, Stallman founded the Free Software Foundation in October 1985. It is "a tax-exempt charity that raises funds to promote the freedom to share and change software. And in the 1980s, one of the main things we did with our funds was to hire people to write parts of GNU. And essential programs, such as the shell and the C library were written this way, as well as parts of other programs."[9] While the Free Software Foundation accepts donations, "most of its income has always come from sales—of copies of free software, and of other related services."[10]

By 1991, Stallman and his programmers had written everything for GNU except the kernel, the part that ties the entire system together. By that time, Linus Torvalds had released the Linux kernel, and he and others combined it with the rest of the GNU operating system (see section on Linus Torvalds and Linux). Almost invariably this operating system has ever since been referred to as Linux. Stallman argues, and some evidence demonstrates, that it should more correctly be called GNU-Linux.

Stallman argues that free software is very much in the interest of commerce by giving businesses control over what the software does or does not do. Businesses that need additional functions can hire programmers to add features if the required skills are not available in-house. In addition, support can be handled the same way. It also ensures privacy and security for the business. "[W]hen a program is proprietary, you can't even tell what it really does . . . it might have a backdoor to let the developer get into your machine. It might snoop on what you do and send information back. This is not unusual."[11]

According to Stallman the worst threat to the free/OSS community comes from the use of software patents instead of copyright as a means of protecting intellectual property rights. Currently, software patents are held and enforced on the compression algorithms that make GIF and MP3 formats possible. Both are widely used formats on the Web, but the patents are often invisible to end users who do not need to pay license fees to view or listen to GIF and MP3 files. However, software developers are required to pay license fees when developing software that can be used to create these files—even if that software is not distributed for a profit. As a result, a creator of an open source package that produces GIF or MP3 files can be sued (as has been threatened to such authors by patent holders). Therefore, the use of the software patent mechanism effectively prevents the creation of OSS.

Berkeley Software Distribution (BSD)

What was once Bell Labs' UNIX operating system has evolved, after a convoluted path, to become another presence in the open source world as three different operating system products: NetBSD, FreeBSD, and OpenBSD. While these operating systems are lss well known than

Linux, their development illustrates how OSS can work and influence other projects.

The University of California at Berkeley obtained a copy of UNIX from Bell Labs in 1974. Over the following four years, Bell Labs and Berkeley enjoyed a strong collaborative relationship that helped UNIX to flourish.[12] However, by 1977 this collaboration also resulted in two distinct branches of the development tree: the Bell Labs UNIX and BSD. BSD was shared with research universities around the world, provided they first purchased a source license from AT&T and with that obtained the full source code for both Bell Labs UNIX and BSD. This model encouraged others to view source code and contribute to its development.

Bell Labs released its final version of UNIX in 1978; "thereafter all UNIX releases from AT&T . . . were managed by a different group that emphasized stable commercial releases. With the commercialization of UNIX, the researchers at Bell Laboratories were no longer able to act as a clearing-house for the ongoing UNIX research."[13] Nevertheless, the research community continued to develop UNIX. As a result, the Berkeley Computer Systems Research Group (CSRG) was formed to replace Bell Labs as an organization that could coordinate and produce further research UNIX releases.

In the early 1980s, the CSRG made several significant additions to UNIX. Key among these was the addition of Internet protocols (TCP/IP). This implementation of TCP/IP has served as the basis of, and sometimes the direct source for, every implementation of TCP/IP since. Eventually, the Berkeley improvements were incorporated in AT&T UNIX (post-Bell Labs versions of UNIX released by AT&T), but for several years, TCP/IP was only available using BSD.

The cost of the required AT&T source license was a prohibitive $50,000 for vendors who "wanted to build standalone TCP/IP-based networking products for the PC market . . . [s]o, they requested that Berkeley break out the networking code and utilities and provide them under licensing terms that did not require an AT&T source license."[14] This was possible because TCP/IP had never been in the Bell Labs' source code and was developed entirely by Berkeley and contributors to BSD. In June 1989, the first freely redistributable source code from Berkeley was released as Networking Release 1.[15] Marshall Kirk McKusick describes its early distribution:

> The licensing terms were liberal. A licensee could release the code modified or unmodified in source or binary form with no accounting or royalties to Berkeley. The only requirements were that the copyright notices in the source file be left intact and that products that incorporated the code indicate in their documentation that the product contained code from the University of California and its contributors. Although Berkeley charged a $1,000 fee to get a tape, anyone was free to get a copy from anyone who already had received it. Indeed, several large sites put it up for anonymous FTP shortly after it was released. Given that it was so easily available, the CSRG was pleased that several hundred organizations purchased copies, since their fees helped fund further development.[16]

One member of CSRG, Keith Bostic, noted the popularity of Networking Release 1 and raised the possibility of producing an expanded release that would include more BSD code. It was pointed out to him that producing such a release would involve replacing hundreds of files originally developed by Bell Labs. Bostic's approach to solving this problem would become the classic model employed elsewhere: using the Internet, he solicited interested programmers to rewrite UNIX utilities based on the published descriptions of those utilities. The only compensation these volunteer programmers would receive would be their name listed by the utility they rewrote in the list of Berkeley contributors. In less than two years, most of the necessary files had been rewritten.

Bostic and two other CSRG members, Marshall Kirk McKusick and Mike Karels, then spent several months reviewing every file in the distribution. Ultimately they determined that six kernel files remained with Bell Labs code that could not quickly be rewritten. Rather than take the time to rewrite those files, and to avoid the delay of drafting a new license agreement, CSRG released the BSD source code they had as Networking Release 2, with the same terms as Networking Release 1. Again, several hundred organizations paid $1,000 for copies of the distribution.

Another CSRG member, Bill Jolitz, incorporated his own files with the Networking Release 2 distribution and released 386BSD. By several accounts, it was not a very stable release, and Jolitz was so particular about the direction of 386BSD that he apparently alienated many of its enthusiasts. Nevertheless, he took the step of making the source code available via anonymous FTP. When he did not make a practice of incorporating contributed bug fixes or producing alternative fixes, a number of 386BSD users formed the NetBSD Group to coordinate the development of this system, and their work became known as NetBSD. This group has always focused on making BSD run on "a large number of hardware platforms."[17] The NetBSD Group, in the tradition of CSRG, has also focused on experimentation and research rather than developing the most stable BSD possible. More information is available at www.netbsd.org.

Another group initially focused continued development exclusively on the Intel x86 platform (it now supports Alpha hardware as well); a few months later this group adopted as its name the FreeBSD Project. FreeBSD has become the most widely used BSD, as its managers have focused on inexpensive CD-ROM distribution and ease of installation. The distribution also includes a Linux emulation package that allows Linux programs to run on FreeBSD machines, as well as access to thousands of open source packages through the "Ports Collection," which allows relatively simple compiling and installation of these packages from source code. More information is available at www.freebsd.org.

From 1994 to 1995, a series of disagreements led to the split of OpenBSD from NetBSD. Theo de Raadt has led development of OpenBSD, focusing on stable, secure distributions which incorporate cryptography. More information is available at www.openbsd.org.

At the same time 386BSD was being developed in 1991 and 1992, Berkeley Software Design began to market a commercially supported version (originally called BSD/386, now called BSD/OS) by writing their own replacement kernel files as Jolitz had. AT&T's subsidiary, UNIX System Laboratories, filed a lawsuit to stop the company from marketing a product that was implied to be UNIX. Eventually, UNIX System Laboratories was sold to Novell. In January 1994, the case was settled out of court and distribution of the source code of the latest version of BSD, called 4.4BSD-Lite, was allowed. Replacing Network Release 2 files with 4.4 BSD-Lite involved another major rewrite of all BSD operating systems.[18]

Another result is that BSD cannot legally be called UNIX, since UNIX is now a registered trademark of the Open Group. However, at least one writer bluntly states, "[h]istorically and technically, it has greater rights than UNIX System V to be called UNIX."[19]

A significant difference between the BSD license and the GNU GPL is that the BSD license does nothing to prevent the creation of proprietary software packages based on modified BSD code. In fact, Microsoft has repeatedly used FreeBSD code implementing TCP/IP in several versions of Windows, and admitted to doing so even as it was criticizing OSS in June 2001.[20] According to Jordan Hubbard, FreeBSD project cofounder, "Rather than take the approach that corporations and other interested parties should be 'forced' into cooperating with the Open Source movement, the many commercial and noncommercial software developers who are behind the BSD movement . . . want their software used by anyone and everyone," with a concern that the GNU GPL will prevent commercial developers from participating at all in open source.[21]

Linus Torvalds and Linux

In October 1991 an undergraduate student at the University of Finland named Linus Torvalds released Linux kernel version 0.02. In announcing its release, he posted this message to the comp.os.minix newsgroup:

> Do you pine for the nice days of Minix-1.1, when men were men and wrote their own device drivers? Are you without a nice project and just dying to cut your teeth on an OS you can try to modify for your own needs? Are you finding it frustrating when everything works on Minix? No more all-nighters to get a nifty program working? Then this post might just be for you.
>
> As I mentioned a month ago, I'm working on a free version of a Minix-lookalike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02 . . . but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it.[22]

Even readers who do not understand the technical terminology can recognize the attitude of wanting to take control of a software project, even at the risk of failure, and the joy of working at an operating system just for the sake of working at it in the hacker tradition.

At the time Torvalds was working on this, BSD source code was not quite fully available, and the GNU kernel, HURD, was mired in development that would eventually take years. But rather than write a complete operating system, Torvalds was already integrating GNU tools with his kernel. It is on this basis that Richard Stallman has argued that Linux should more properly be called GNU-Linux. However, Marshall Kirk McKusick has pointed out that, "about half of the utilities that [Linux] comes packaged with are drawn from the BSD distribution."[23]

Torvalds focuses on the technology and the community that built it:

> Linux today has millions of users, thousands of developers, and a growing market. It is used in embedded systems; it is used to control robotic devices; it has flown on the space shuttle. I'd like to say that I knew this would happen, that it's all part of the plan for world domination. But honestly this has all taken me a bit by surprise. I was much more aware of the transition from one Linux user to one hundred Linux users than the transition from one hundred to one million users.[24]

This last statement bears emphasis and further examination, as it belies a development style somewhat different from those of GNU, BSD, and Apache. These packages were developed "in a carefully coordinated way by a small, tightly-knit group of people . . . [while Linux, by comparison], was rather casually hacked on by huge numbers of volunteers coordinating over the Internet."[25] Raymond describes these approaches to development as "Cathedral" and "Bazaar," respectively:

> Quality [in the "Bazaar" model] was maintained not by rigid standards or autocracy but by the naively simple

strategy of releasing every week and getting feedback from hundreds of users within days, creating a sort of rapid Darwinian selection on the mutations introduced by developers. To the amazement of almost everyone, this worked quite well.[26]

Eric Raymond further asserts, "[b]y late 1993, Linux could compete on stability with many commercial UNIXs, and it hosted vastly more software. It was even beginning to attract ports of commercial applications software."[27] This trend has only continued. According to Raymond:

> Linux is a project that was conceived some five years after Microsoft began development of Windows NT. Microsoft has spent tens of thousands of man-hours and millions of dollars on the development of Windows NT. Yet today Linux is considered a competitive alternative to NT as a PC-based server system, an alternative that major middleware and backend software is being ported to by Oracle, IBM, and other major providers of enterprise software. The Open Source development model has produced a piece of software that would otherwise require the might and resources of someone like Microsoft to create.[28]

Over the past five years, the computer trade press has greatly increased coverage of OSS, primarily Linux. The question for many information technology (IT) professionals has changed. Instead of asking themselves if they will use OSS in their shops, they are now asking where they will use it.

Eric Raymond's collection of essays published in *The Cathedral and the Bazaar* has examined the reasons for Linux's success in great depth. In fact, Raymond deliberately imitated Torvalds's development style as an experiment when managing the fetchmail project and analyzed it in the essay of the same name.[29] A frequently quoted message from that essay is, "Given enough eyeballs, all bugs are shallow."[30] Torvalds extends this by again focusing on the Linux community:

> The power of Linux is as much about the community of cooperation behind it as the code itself. If Linux were hijacked—if someone attempted to make and distribute a proprietary version—the appeal of Linux, which is essentially the open-source development model, would be lost for that proprietary version.[31]

Beyond the stability of the project, Linux has also prompted the development of a business model that might not seem possible: selling freely available software and making a profit. By 1994, a number of distributions of Linux were available for less than thirty dollars. These included Yggdrasil, Slackware, Debian, Suse, and others. One, Red Hat, has grown into a publicly traded company. According to Robert Young, CEO of Red Hat, "you make money in free software exactly the same way you do it in proprietary software: by building a great product, marketing it with skill and imagination, looking after your customers, and thereby building a brand that stands for quality and customer service."[32]

Other Widely Used Open Source Packages

Thus far this article has focused on the development of open source operating systems and related tools. But in fact a number of software packages have been developed as open source projects. Among them:

- In 1987 Larry Wall released PERL 1.0 scripting/programming language. Its current release is version 5.6.
- In 1990 Guido van Rossum released Python programming language.
- In 1994 Rasmus Lerdorf released PHP/FI Web scripting/programming language. Its current release is PHP 4.0.6.
- In 1995 the Apache Web server program was released and quickly became the most widely used Web server product (which it remains today).
- In the mid-1990s mSQL, MySQL, and PostgreSQL relational databases were released.
- Also in the mid-1990s Andrew Tridgell released Samba, a set of utilities that allows UNIX machines to use the same network communication protocol as Microsoft Windows.

The developers of these packages have focused on the cultures surrounding the projects as well as the projects themselves. Larry Wall has summed up this emphasis by saying, "As a linguist, I understood that a language without a culture is dead. If you get the culture right, the technology will happen."[33]

While many of these packages originally ran exclusively on UNIX, most have been ported to other operating systems, including Windows. The other significant development was the involvement of commercial interests in the open source movement. As already stated, corporations such as IBM and Oracle have ported software to the Linux platform. Just as significantly, commercial training, certification, and support are available for Apache, MySQL, and PostgreSQL, just as they are for Linux.

In January 1998 Netscape released the source code for its browser under an open source license, beginning the Mozilla project. As of November 2001 Mozilla had not yet produced a release version 1.0, but was at 0.9.4. This might seem to classify Mozilla as less than successful, but two issues argue against that idea:

- While development has been relatively slow, it has been steady, and recent versions have been quite stable.
- The less visible issue in 1998 was that Netscape made most of its money selling server software. "For Netscape the issue was less about browser-related

income (never more than a small fraction of their revenues) than maintaining a safe space for their much more valuable server business. If Microsoft's Internet Explorer achieved market dominance, Microsoft would be able to bend the Web's protocols away from open standards and into proprietary channels that only *Microsoft*'s servers would be able to service."[34]

Development of the Term "Open Source"

Considering that what is now characterized as the open source movement has been in conscious development for nearly two decades, the term "open source" itself has been a relative latecomer. In fact, the term was proposed and voted on by a group of people who were meeting on a regular basis in late 1997 and early 1998 and who were interested in spreading awareness of the sophisticated tools that had been developed outside the proprietary software development model. The term was proposed by Christine Peterson of the Foresight Institute. Peterson tells the story:

> It was a very deliberate effort at a name change. In late 1997 or very early 1998, a number of us felt the name "free software" was holding back the budding industry/movement. Newcomers always thought "free" meant free-as-in-beer, not free-as-in-speech. Various ideas were kicked around, including at a meeting at Foresight, but none were catching on.
>
> On my own I thought of open source, ran it by my PR friend (who didn't like it) and a couple of personal friends (who did).
>
> At the next meeting of the little group convened by Eric Raymond, this time at VA, I was shy about suggesting the new term—I had no standing with this group. So Todd Andersen, with whom I'd planned the name change effort, just used it casually, not suggesting it as an explicit new term. It caught on immediately at the meeting, without the others noticing except Todd and me, who winked at each other. At the end of the meeting, it was pointed out by Todd or me that this new term seemed to be working, and people seemed willing to give it a try.
>
> The others later weren't sure who came up with it, but Todd made sure I got credit, which I usually don't push for. Nice of him. And Eric Raymond has also been good about making sure I get credit for it.
>
> Now I try to go around renaming things all the time.[35]

Eric Raymond remembers the meeting a bit differently: "I remember hearing Christine say 'open source' and thinking 'we have a winner.' It may have slipped by other people, but it didn't get by me."[36]

This group registered the domain name opensource.org, defined "open source," developed OSI certification, and created a list of licenses that meet the standards for open source certification.

The basic definition is:

- The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources.
- The program must include source code and must allow distribution in source code as well as compiled form.
- The license must allow modifications and derived works and must allow them to be distributed under the same terms as the license of the original software.
- The license may restrict source code from being distributed in modified form only if the license allows the distribution of patch files with the source code for the purpose of modifying the program at build time.
- The license must not discriminate against any person or group of persons.
- The license must not restrict anyone from making use of the program in a specific field of endeavor.
- The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
- The license must not be specific to a product.
- The license must not contaminate other software by placing restrictions on any software distributed along with the licensed software.[37]

"The Open Source Definition allows greater liberties with licensing than the GPL does. In particular, the Open Source Definition allows greater promiscuity when mixing proprietary and open-source software."[38] This is Richard Stallman's objection to OSS—that it allows the inclusion of proprietary software and ignores the philosophical issue of software freedom. Without these freedoms, there is no philosophical imperative to improve one's community. Nevertheless, "[w]e disagree on the basic principles, but agree more or less on the practical recommendations. So we can and do work together on many specific projects. We don't think of the Open Source movement as the enemy."[39]

This is a point reiterated by many who are active in various competing open source and free software packages. While this article has focused on a number of differences between operating systems, approaches to collaboration, and the evolution of various license agreements, this focus is at the micro level. At the macro level, nearly everyone mentioned in this article would prefer a competing open source or free package to a proprietary software package. In the future those who have blazed new trails will continue to argue the finer distinctions between their respective works. However, the various groups involved are willing to work with and support one another's right to choose a different approach to solving a problem. And it is clear these individuals look forward to another generation building upon the successes of the past thirty years.

References and Notes

1. Eric S. Raymond, "A Brief History of Hackerdom," in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (Sebastopol, Calif.: O'Reilly and Assoc., 1999). Also available at http://tuxedo.org/~esr/writings/cathedral-bazaar/hacker-history. Accessed Oct. 20, 2001.

2. Eric S. Raymond, "Homesteading the Noosphere," in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (Sebastopol, Calif.: O'Reilly and Assoc., 1999). Also available at http://tuxedo.org/~esr/writings/cathedral-bazaar/hacker-history. Accessed Oct. 20, 2001.

3. Richard M. Stallman, Transcript of Richard M. Stallman's speech, "Free Software: Freedom and Cooperation," New York University, New York, May 29, 2001 in GNU's Not Unix! Accessed Aug. 23, 2001, www.gnu.org/events/rms-nyu-2001-transcript.txt.

4. Ibid.

5. Ibid.

6. Richard M. Stallman, "The GNU Operating System and the Free Software Movement," in *Open Sources: Voices from the Open Source Revolution,* edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, Calif.: O'Reilly and Assoc., 1999), 56.

7. Stallman, "Free Software: Freedom and Cooperation."

8. "What Is CopyLeft?," in *Free Software Licenses—GNU Project,* Sept. 15, 2001. Accessed Oct. 22, 2001, www.gnu.org/licenses/licenses.html.

9. Stallman, "Free Software: Freedom and Cooperation."

10. Stallman, "The GNU Operating System," 60.

11. Stallman, "Free Software: Freedom and Cooperation."

12. This evolution is described in careful detail in Marshall Kirk McKusick, "Twenty Years of Berkeley UNIX: From AT&T-Owned to Freely Redistributable" in *Open Sources: Voices from the Open Source Revolution,* edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, Calif.: O'Reilly and Assoc., 1999), 31–46. Much of this section is condensed from that source. Also, a timeline depicting the entire history of AT&T UNIX and BSD is available at ftp://ftp.netbsd.org/pub/NetBSD/NetBSD-current/src/share/mis/bsd-family-tree. Accessed Oct. 20, 2002.

13. McKusick, "Twenty Years of Berkeley UNIX," 34–35.

14. Greg Lehey, The Daemon's Advocate: Anarchies, Monarchies, and Dictatorships, in Daemonnews: Bringing BSD Together, Oct. 2000. Accessed Nov. 11, 2001, http://daemonnews.org/200010/dadvocate.html.

15. McKusick, "Twenty Years of Berkeley UNIX," 40.

16. Ibid., 41.

17. NetBSD, About NetBSD. Accessed Sept. 2, 2001, www.netbsd.org/Misc/about.html.

18. Jordan Hubbard, "A Brief History of FreeBSD," in FreeBSD Handbook. Accessed Nov. 10, 2001, www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/index.html.

19. Greg Lehey, introduction to *The Complete FreeBSD*, 3d ed. (Walnut Creek, Calif.: Walnut Creek CD-ROM, 1999), xxix.

20. Lee Gomes, "E-Business: Microsoft Uses Free Code," *The Wall Street Journal*, June 18, 2001.

21. Jordan Hubbard, "Mister, How Far Is Licensing from Utopia?" *Open*, 2.6 (June 2001), 48.

22. Linus Torvalds, Linux History, in Linux International, ©2001. Accessed Nov. 10, 2001, www.li.org/linuxhistory.php. Torvalds's entire series of announcements is available at this site.

23. McKusick, "Twenty Years of Berkeley UNIX," 46.

24. Linus Torvalds, "The Linux Edge," in *Open Sources: Voices from the Open Source Revolution*, edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, Calif.: O'Reilly and Assoc., 1999), 101.

25. Raymond, "A Brief History of Hackerdom," 24.

26. Ibid.

27. Ibid.

28. Chris DiBona, Sam Ockman, and Mark Stone, eds., introduction to *Open Sources: Voices from the Open Source Revolution,* (Sebastopol, Calif.: O'Reilly and Assoc., 1999), 17.

29. Eric S. Raymond, The fetchmail Home Page, Nov. 08, 2001. Accessed Nov. 11, 2001, http://tuxedo.org/~esr/fetchmail.

30. Eric S. Raymond, "The Cathedral and the Bazaar," in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (Sebastopol, Calif.: O'Reilly and Assoc., 1999), 41.

31. Torvalds, "The Linux Edge," 109.

32. Robert Young, "Giving It Away: How Red Hat Software Stumbled across a New Economic Model and Helped Improve an Industry," in *Open Sources: Voices from the Open Source Revolution,* edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, Calif.: O'Reilly and Assoc., 1999), 114.

33. Larry Wall, "The History of PERL," presentation delivered at the 120th ALA Annual Conference, San Francisco, Calif., June 17, 2001, as part of the program, "Web Tools and Digital Resources: Open Source Then and Now."

34. Eric S. Raymond, "The Revenge of the Hackers," in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (Sebastopol, Calif.: O'Reilly and Assoc., 1999), 202.

35. Christine Peterson, "Quick Question about the Term 'Open Source,'" personal e-mail to the author, Aug. 31, 2001.

36. Eric S. Raymond, "Quick Question about the Term 'Open Source,'" personal e-mail to the author, Sept. 1, 2001.

37. OpenSource.org., The Open Source Definition, Version 1.8. Accessed Sept. 1, 2001, http://opensource.org/docs/definition.html.

38. DiBona, Ockman, and Stone, eds., introduction to *Open Sources*, 3.

39. Why "Free Software" Is Better than "Open Source," in GNU Project—Free Software Foundation, Aug. 20, 2001. Accessed Oct. 22, 2001, www.gnu.org/philosophy/free-software-for-freedom.html.

Find it. Faster.

You just got a request for a list of grocery stores in a three-state area with more than 20 employees.

You could search through dozens of reference sources to create that list. Or, you could call the Library Division of *info*USA.

We offer the country's most extensive databases of business and residential information. And, you choose the format you want: print, CD-ROM or via the Internet.

Want more information? Call us today at **1-800-808-1113** or e-mail: **library@infoUSA.com.**

Your search is over.

*info*USA

Library Division

5711 S. 86th Circle • P.O. Box 27347 • Omaha, NE 68127
Phone: (402) 593-4523 • Fax: (402) 596-7688 • www.libraryUSA.com

23930

Possibilities for Open Source Software in Libraries

Eric Lease Morgan

This short essay, based on a presentation given at the 2001 American Library Association (ALA) Annual Conference, enumerates a number of possibilities for open source software (OSS) in libraries and how it can be leveraged to provide better and more effective digital library collections and services.

OSS Briefly Defined

Open source software (OSS) is both a philosophy and a process. It is a philosophy describing the intended use of software and methods of distribution. OSS is often times equated with GNU software as well as described as free software, but the term "free" should be more equated with the Latin word *liberat* (meaning to liberate), and not necessarily *gratis* (meaning without return made or expected). In the words of Richard Stallman, the founder of the GNU software project, we should "think of 'free' as in 'free speech,' not as in 'free beer.'"[1] In this regard, the ideology behind OSS is not unlike some of the basic principles of librarianship in America.[2]

OSS is also a process for the creation and maintenance of software. This is not a formalized process, but rather a process of convention with common characteristics between software projects. First and foremost, the developer of a software project almost always is trying to solve a specific computer problem—commonly called "scratching an itch." The developer realizes other people may have the same problem, and consequently the developer makes the project's source code available on the Internet in the hopes that other people can use it too. If there seems to be a common need for the software, a mailing list is usually created to facilitate communication, and hopefully the list is archived. Since the software is almost always in a state of flux, developers need some sort of version-control software to help manage the project's components. The most common version-control software is called CVS (Concurrent Versions System).

Codevelopers then "hack away" at the project, adding features they desire or fixing bugs of previous releases. As these features and fixes are created, the source code's modifications, in the form of "diff" files, are sent back to the project's leader. The leader examines the diff files, assesses their value, and decides whether or not to include them into the master archive. The cycle then begins anew. Much of a project's success relies on the primary developer's ability to foster communication and a sense of community around the project. Once accomplished, the "two heads are better than one" philosophy takes effect and the project matures. A highly recommended book titled *The Cathedral and the Bazaar* by Eric S. Raymond outlines this process in much greater detail.[3]

OSS Contrasted with Homegrown Systems

Some people may remember the homegrown integrated library systems developed in the '70s and '80s, and these same people may wonder how OSS is different from those humble beginnings. There are two distinct differences. The first is the present-day existence of the Internet. This global network of computers enables people to communicate over much greater distances, and it is much less expensive than twenty-five years ago. Consequently, developers are not as isolated as they once were and the flow of ideas travels more easily between developers—people who are trying to scratch an itch. Yes, there were telephone lines and modems, but the processes for using them were not as seamlessly integrated into the computing environment (and there were always long-distance communications charges to contend with).[4]

Second, the state of computer technology and its availability has dramatically increased in the past twenty-five years. At that time, computers, especially the type used for large-scale library operations, were almost always physically large, extremely expensive, remote devices whose access was limited to a small group of specialized individuals. Today, the computers on most people's desktops have enough RAM, CPU horsepower, and disk space to support a college campus of twenty-five years ago.[5]

In short, the OSS development process is not like the homegrown library systems of the past simply because there are more people with more computers who are able to examine and explore the possibilities of solving more computing problems. In the days of the homegrown systems people were more isolated in their development efforts and more limited in their choice of computing hardware and software resources.

State of OSS in Libraries

What is the state of OSS in libraries today? Daniel Chudnov has been the profession's evangelist for the past two or three years, the original author of jake (jointly administered knowledge environment), and the main-

Eric Lease Morgan (emorgan@nd.edu) is Head of the Digital Access and Information Architecture Department at the University Libraries of Notre Dame, Indiana.

tainer of the oss4lib.org domain as well as its mailing list. Chudov has done a lot to raise the awareness of OSS in libraries. To that end he, Gillian Mayman, and others maintain a list of open source system projects. These projects include a lot of software designed specifically for libraries such as (but not limited to):

- Document delivery applications (Prospero by Eric Schnell)
- Z39.50 clients and servers (Yaz and SimpleServer by Sebastian Hammer, Zeta Perl by Rocco Carbone, and JZKit by Knowledge Integration, Ltd.)
- Systems to manage collections (Catalog by Senga, Greenstone by Ian H. Witten et al., ROADS funded by JISC via the eLib Programme, and OSCR by Wally Grotophorst)
- MARC record readers and writers (MARC.pm by Bearden et al., m[n]m by Robert McDonald et al., and XMLMARC by Lane Medical Library)
- Integrated library systems (Avanti by Peter Schlumpf, Koha by Rosalie Blake and Rachel Hamilton-Williams, OpenBook by the Technology Resource Foundation, and OSDLSP by Jeremy Frumkin and Art Rhyno)
- Systems to read and write bibliographies (bib2html by Stephanie Galland, bp by Dana Jacobsen, gBib by Alejandro Sierra and Felipe Bergo, and Pybliographer by Frederic Gobry)

For a more comprehensive list, visit www.oss4lib.org.

Yet the state of OSS in libraries is more than sets of computer programs. It also includes the environment where the software is intended to be used—a socioeconomic infrastructure. Put another way, any computing problem can roughly be divided into 20 percent technology issues and 80 percent people issues. It is this 80 percent of the problem that concerns us here. Given the current networked environment, the affinity of OSS development to librarianship, and the sorts of projects enumerated above, what can the library profession do to best take advantage of the current milieu? This question was posed to the OSS4Lib mailing list in April and May of 2000 and generated a lively discussion.[6] A number of themes presented themselves, each of which are elaborated upon below:

- National leadership
- Mainstreaming, workshops, and training
- Usability and packaging
- Economic viability
- Redefining the integrated library system (ILS)
- Open source data

National Leadership

One of the strongest themes mentioned was the need for national leadership. It was first articulated by David Dorman as the Open Source Library Network (OSLN). Karen Coyle and Aaron Trehab elaborated on the idea by suggesting that organizations such as ALA/LITA, DLF, OCLC, or RLG help fund and facilitate methods for providing credibility, publicity, stability, and coordination to library-based OSS projects. While OSS is almost always driven by individuals, the individuals of OSS still need to be provided with resources such as time, money, and computer hardware and software. It is widely believed that individualism can only go so far because after a time, individuals lose interest and pass projects on to others. Libraries are in it for the long term and cannot afford to implement workflows based on software whose lifetime is measured in "Internet years." National leadership, in the form of institutionalized support, will make OSS in libraries more of a reality much in the same way RedHat has helped make Linux a viable operating system and the World Wide Web Consortium (W3C), supported by MIT, provides guidelines and standards for the Web.

Mainstreaming, Workshops, and Training

Along these same lines was the expressed desire for the mainstreaming of OSS articulated by Carol Erkens, Rachel Cheng, and Peter Schlumpf. This mainstreaming process would include presentations, workshops, and training sessions on local, regional, and national levels. These activities would describe and demonstrate OSS for libraries. They would enumerate the advantages and disadvantages of OSS. They would provide extensive instruction on the staffing, installation, and maintenance issues of OSS. This mainstreaming process is an effort to promote and market OSS as a viable means for implementing sustainable digital library collections and services.

Usability and Packaging

In its present state, OSS is much like microcomputer computing of the '70s as stated by Blake Carver. It is very much a build-it-yourself enterprise; the systems are not very usable when it comes to installation. This point was echoed by Cheng, who helped facilitate a NERCOMP workshop on OSS. Schulmpf pointed to the need for easier installation methods so maintainers of systems can focus on managing content and not software. Using OSS should not be like owning an automobile in the '20s; you shouldn't necessarily need to know how to fix it in order to make it go. Packaging, and to a lesser extent, usability, are features supported in software by commercial institutions. Again, RedHat, a company distributing versions of the Linux operating system, has made its money by making it easier to install and maintain Linux-based computers. Microsoft writes software intended to seamlessly integrate with Intel-based computers. Microsoft's success is not based so much on the features of its applications, but rather the way the applications integrate with each other. The developers of OSS, including the ones in libraries, would benefit from similar installation procedures and integration processes.

Economic Viability

As pointed out by Eric Schnell and David Dorman, OSS needs to be demonstrated as an economically viable method of supporting software and systems. Libraries have spent a lot of time, effort, and money on resource sharing. Why not pool these same resources together to create software that will satisfy our professional needs? OSS cannot be equated with the homegrown systems of the past—spaghetti code and GOTO statements should be ancient history. More importantly, a globally networked computer environment provides a means of sharing expertise in a manner not feasible twenty-five years ago. We need to demonstrate to administrators and funding sources that money spent developing software empowers our collective whole. It is an investment in personnel and infrastructure. OSS is not a fad, yet it will not necessarily become a complete replacement for commercial software. On the other hand, OSS offers opportunities not necessarily available from the commercial sector.

Redefining the ILS

There are many open source library applications available today, and each satisfies a particular need. Maybe each of these individual applications can be brought together into a collective, synergistic whole as described by Jeremy Frumkin, and we could redefine the ILS. Presently our ILSs manage things like books pretty well. With the addition of 856 fields in MARC records they are beginning to assist in the management of networked resources too, but libraries are more than books and networked resources. Libraries are also about services: reserves, reading lists, bibliographies, reader advisory services of many types, digitization, current awareness, reference, to name but a few. Maybe the existing OSS can be glued together to form something more holistic—a sum greater than its parts.

OSS provides an opportunity for traditional library vendors as described by Schnell. Instead of writing computer programs, library vendors could support the documentation, installation, and integration of OSS for libraries in exchange for a fee. Libraries would feel much more comfortable with the applications running on their computers if those applications did not seem to so much beyond their control.

Open Source Data

OSS relates to data as well as systems, as described by Thomas Krichel. The globally networked computer environment allows us to share data as well as software. Why not selectively feed URLs to Internet spiders to create our own, subject-specific indexes? Why not institutionalize services like the Open Directory Project or build on the strength of INFOMINE to share records in a manner similar to the manner of OCLC?

Systematically describing Internet-accessible information resources with things like the Resource Discovery Framework (RDF) provides the means of implementing the Semantic Web. Libraries are about the collection, organization, dissemination, and evaluation of data and information for the purposes of facilitating knowledge. These are the same principles behind the Semantic Web—a tool for answering the perennial question, "Can you find me more like this one?" The library profession purports to excel at the classification of data and information. RDF represents one way to accomplish this goal in a globally networked environment. If we, as librarians, were to contribute to the efforts of the Semantic Web, then we would also be contributing to the efforts of open source data.[7]

Another way to contribute to the open source data concept is to integrate ourselves into information creation processes of our hosting communities. Libraries do not exist in a vacuum. They are all a part of some sort of community. Each of these communities creates information and increasingly makes it available in digital form. By becoming a part of this process libraries may be able to make the information more accessible to a wider audience over a longer period of time. Again, this is fostering a concept of open source data.

Possibilities of OSS in Libraries

OSS presents many possibilities for libraries. First and foremost, it presents an opportunity to take control of library services and collections relying on computer software. The time and effort spent buying (read "licensing") software could be developed learning how to use the software. Time spent developing our own solutions to problems develops staff expertise. OSS lowers the barrier to this learning process because staff will not be limited by such things as who is allowed to use the software; since the software is freely given away, it is very easy to download, install, give it whirl, and evaluate whether or not more time should be spent on it.

Quality OSS rises to the top in the same way cream rises to the top of fresh milk. OSS goes through an informal review process. This process has nothing to do with market hype or self-promotion. Consequently, if you identify a piece of OSS and you know of other people using it, then you will know exactly what you are getting. There should be no surprises. This presents itself as another opportunity for libraries, not so much in terms of library services or collections, but in the time spent evaluating products. In the words of Shiyali Ramarita Ranganathan, "Save the time of the reader."[8]

Instead of feeling helpless about how our online catalogs work (or don't work), or instead of wishing for some sort of software widget to "automagically" appear, OSS provides a framework—possibilities for resource sharing—in order to take control of our situation. We all have similar problems, needs, and desires when it comes to using computers in our libraries. If we were to take a greater stake in the use of OSS, then we would be more able to share our ideas among ourselves. This sharing of ideas will bring more minds together and ultimately create more robust solutions. Effective communication will still have to take place, but that is where the leadership comes into play. OSS does not solve communication problems.

OSS provides the means to give back to the Internet. By contributing OSS to the community at large, others will benefit from our experience. Similarly, if we, as a profession, contribute to the idea of open source data through the systematic description of Internet resources, then we will be helping people satisfy their information needs. We will be bringing like things together for the purposes of creating knowledge, not just gathering information. While information has never been free, the processes behind OSS can make it less expensive.

Conclusion

I am always excited about libraries and librarianship. The discussions on the oss4lib mailing list exemplify some of the opportunities for our profession. As Ben Ostrowsky put it, "[y]ears from now, this will be known as The Week It All Came Together." We can hope so. Let's hope the momentum can be sustained. Let's build on our strengths, continue to pool our resources, and spend our time, money, and energy on ways to improve our situation instead of bemoaning the perceived limitations. As Gordon Paynter said, "These are social problems, rather than technical." Let's explore our alternatives.

References and Notes

1. The ideas behind GNU software and its definition as articulated by Richard Stallman can be found at www.gnu.org/philosophy/free-sw.html. Accessed Jan. 10, 2002.
2. I elaborated on the similarities and differences between OSS and librarianship via a book review of *The Cathedral and the Bazaar* appearing in *Information Technology and Libraries* 19, no. 2 (June 2000): 105. See www.lita.org/ital/1902_books.html#anchor387677. Accessed Jan. 10, 2002.
3. *The Cathedral and the Bazaar* is also available online at www.tuxedo.org/~esr/writings/cathedral-bazaar. Accessed Jan. 10, 2002.
4. As an interesting aside, read "Stalking the Wily Hacker" by Clifford Stoll in the *Communications of the ACM* 31, no. 5 (May 1988): 484. The essay describes how Clifford tracked a hacker via a seventy-five-cent error in his telephone bill. It is on the Web in many places. Try http://eserver.org/cyber/stoll2.txt. Accessed Jan. 10, 2002.
5. It is believed a past chairman of IBM, Thomas Watson, said in 1943, "I think there is a world market for maybe five computers."
6. An archive of the oss4lib mailing list is available at www.geocrawler.com/lists/3/SourceForge/6067/0. Accessed Jan. 10, 2002.
7. I personally think the ideas behind the Semantic Web are very intriguing. For more information about this effort see www.w3.org/2001/sw. Accessed Jan. 10, 2002.
8. Ranganathan's Five Laws of Library Science are: (1) books are for use; (2) every book its reader; (3) every reader his book; (4) save the time of the reader; and (5) a library is a growing organism.

The Open Source ILS: Still Only a Distant Possibility

Marshall Breeding

One of my main professional interests involves following the library automation industry. I maintain a Web site devoted to this topic and regularly write about the companies and systems that comprise this arena. So it is with great interest that I consider the impact open source software (OSS) might make on this industry. The open source movement could effect radical changes to libraries should it produce an integrated library system (ILS) that earns a level of acceptance on the same order that Apache did in the Web server market. Like Apache, an open source ILS would have to offer top-of-the-line features and performance to gain acceptance over its commercial rivals.

My general approach to software, technologies, and systems is initial skepticism. I've learned that the hype about any new technology usually exceeds its practical impact in the long term. My attitude toward OSS in libraries is no different. While I appreciate its successes, I also recognize its limitations. There is no doubt that Linux and Apache represent a worldwide victory over high-powered commercial opponents in the operating system and Web server arenas.[1]

I do not, however, expect to see such victories of OSS over commercial products in the ILS arena. Both broad historical and recent trends argue against a movement toward libraries creating their own library automation systems—either in an open source or closed development process.

An undeniable trend in library automation involves a movement toward vendor-supplied systems and away from locally developed ones. Libraries large and small recognize that they do not have the resources to develop and maintain library automation systems. Some of the recent examples that come to mind include:

- Library of Congress: Adopted Endeavor Voyager to replace several locally developed systems.
- UCLA: Early implemented DRA Taos over their locally developed ORION system.
- Stanford University: Abandoned locally developed BALLOTS system for SIRSI Unicorn.
- Penn State: Converted from their locally developed LIAS system to SIRSI Unicorn.

Other less prominent libraries that have left locally developed systems for commercial systems include Carelton University, Jefferson County (Colo.) Public Library System, and the Fogelson Library of the College of Santa Fe.

Marshall Breeding (breeding@library.vanderbilt.edu) is Library Technology Officer for the Heard Library at Vanderbilt University, Nashville, Tenn.

Very few large libraries continue to operate locally developed library automation systems. The only two ARL libraries currently running locally developed systems are the University of California at Berkeley and the University of Texas at Austin. Neither of these locally developed systems is open source. It should be noted, however, that the development of these systems far predates the open source movement. One may well speculate that had the open source movement been in place during the period in which many libraries were creating library automation systems, the current environment of reliance on commercial systems would be quite different.

The complexity of library automations systems exceeds the pool of available volunteer programmers. Full-fledged ILS software can easily contain a million lines of software code. Library automation companies that have recently undertaken the development of a new ILS have generally expended about five years of development time with a team of thirty to fifty programmers. The creation of a new ILS is a multimillion-dollar project. It is hard to see that even a large collective of libraries would have the available programming staff to develop and maintain a large-scale ILS.

Preferred technology architectures evolve faster than the development cycles of applications built upon them. The history of technology has seen constant shifts in computing models and architectures. Time-sharing host/terminal systems gave way to client/server systems with thick graphical clients. These gave way to an N-Tier architecture and Web-based clients. Some models prove to be more long lasting than others. The ill-fated Taos system comes to mind as one such example. CORBA and object-oriented databases, the underpinnings of Taos that were in the height of fashion when DRA's Taos system was conceived, do not currently enjoy the same level of respect. Even with massively funded, concerted development efforts by commercial companies, a system is doing well if its technical infrastructure is not obsolete by the time the application is fully developed.

Even with teams of full-time development and support personnel, commercial companies often cannot keep up with their library customer's expectations. Is it realistic to believe that a cadre of open source developers would have the available resources to keep pace with the ever-rising expectations of libraries for system enhancements and ongoing support?

The open source approach would posit that a worldwide collaborative effort would bring about more satisfactory software since it would be built for and by libraries. Yet it is hard to believe that teams of programmers, working in a mostly voluntary mode throughout libraries everywhere, would have the time allotment, project management infrastructure, and other resources needed for the concerted development efforts required to build and maintain an ILS.

Libraries expect far more from their library automation vendors than software. In the current market of relatively mature systems, each of which has achieved a high level of core functionality, differentiating factors between systems include quality of support, the vision of the company regarding broad technology and industry trends, guaranteed continued development of the system relative to evolving requirements, standards, and technology architectures.

While there are numerous open source enthusiasts in the technology and computer support units of libraries, few library administrators have demonstrated interest in taking on the risks and responsibilities of strategic reliance on open source library automation systems. A handful of programmers in libraries currently work on open source projects in small blocks of time either officially or unofficially allocated. But it does not seem likely that library administrators are ready to switch from paying license and support fees to commercial firms to paying for local development and support. Programmers are expensive personnel. Libraries generally lack the ability to fund adequate programming and technical staff in support of commercially supplied systems, much less toward the development of new open source systems.

Most of the current open source ILS projects that I am aware of can be characterized as relatively small-scale efforts. Some rely on paid staff, others depend on volunteers. It is hard to see how a full-fledged scaleable ILS can emerge out of the current projects. The open source ILS projects that I have been tracking include: Avanti (www.nslsilus.org/~schlumpf/avanti), OSDLS/Pytheas (http://osdls.library.arizona.edu), Koha from Katipo Communications (www.katipo.co.nz), and OpenBook from the Technology Resource Foundation (www.trfoundation.org). The latter two systems show the most promise. Yet it is hard to discern any significant trend of even small libraries adopting these systems in favor of the small-scale systems available from commercial vendors. OpenBook, which may show the most promise, is still not ready for release and is behind its initial schedule.

Several technical components that are useful to those that build and integrate library automation systems and other library software are available. Some of these open source utilities include:

- James, a Java API for MARC records;
- YAZ, a Z39.50 toolkit;
- ZAP, an Apache module for Z38.50;
- XMLMARC, utility for converting MARC records to XML; and
- ZETA Perl, a Perl module for Z39.50.

My concluding observation in the ILS arena is that none of the current library automation vendors have expressed concern that their efforts will be usurped by open source efforts. Their customer base continues to grow and they each describe a long-term agenda of future development. Libraries are buying more commercial software, not less. While my idealistic side might like to see libraries able to obtain automation systems without cost in an open source arrangement, my practical observations show little movement in this direction. As one who closely follows the library automation industry, I can see no paradigm shift approaching where commercial companies yield to open source and free software.

While I've argued that an open source ILS does not seem to be a realistic expectation, at least not in the near future, I do see other significant opportunities for libraries to take advantage of the open source model.

It is likely that OSS will impact libraries in non-ILS arenas—which may ultimately be more important. The recent interest in digital library initiatives present opportunities for OSS. The early software for the Open Archives Initiative, for example, is largely OSS. As XML gains ground in the library arena, it is also likely that a significant body of OSS will arise.

We already see some inroads of OSS for libraries. OCLC, for example, recently released the Java source code to its SiteSearch toolkit for noncommercial use. While OCLC determined that SiteSearch could not be sustained in the usual commercial development and support model, it is willing to provide some level of help to the community of librarians who want to build a future for SiteSearch as a community source application (because of the limitation not allowing commercial use of the software, the SiteSearch license is not a true open source license). OCLC also released their Pairs search engine and several other tools under open source licenses, and they appear to be interested in future open source projects.

George Mason University has created the Open Source Course Reserves (OSCR), an electronic course reserves system designed for academic libraries (see http://timesync.gmu.edu/OSCR).

Some commercial companies use open source components within their systems and even contribute open source modules. For example, epixtech indicates that they will develop an NCIP interface in an open source license arrangement.[2]

The opportunity for libraries to develop open source applications in the digital library arena is narrow. One of the major trends among ILS companies involves an interest in creating products with broader information delivery, content integration, and resource sharing. It will be interesting to observe whether open source applications will be developed and survive in these areas that can compete with the emerging commercial offerings.

For a list of resources related to OSS, visit the Library Technology Guides (http://staffweb.library.vanderbilt.edu/breeding/ltg.html), select the bibliography section, and perform a subject search for "open source software."

References and Notes

1. For an article affirming these trends, see David A. Wheeler, Why Open Source Software/Free Software (OSS/FS)? Look at the Numbers! Feb. 1, 2001. Accessed Feb. 17, 2002, www.dwheeler.com/oss_fs_why.html.

2. Epixtech, Epixtech Announces Open Source Licensing Plans for NISO CIP "Engine," Jan. 15, 2001. Accessed Feb. 17, 2002, www.epixtech.com/ls/press/2001/9998.asp.

5th ANNUAL

LITA NATIONAL

FORUM

OCTOBER 10–13, 2002
HOUSTON, TEXAS

MAKING CONNECTIONS

Thoughtful general sessions, preconference workshops, and pragmatic concurrent sessions on forging connections in today's library and information environment.

Visit the forum Web site for developments and details:

www.lita.org/forum02

Registration begins July 2002 and is limited to 500.

a tradition of the
Library and Information Technology Association
A division of the American Library Association

MARC It Your Way: MARC.pm

Anne Highsmith, Mark Jordan, Eileen Llona, Peter E. Murray, and Edward Summers

MARC.pm (http://marcpm.sourceforge.net) is a piece of open source software (OSS) developed by librarians for librarians. In this article you will find a description of what exactly MARC.pm is, followed by a series of descriptive pieces written by librarians in the field who have used MARC.pm. Some of these descriptions contain program code, which may baffle those who are not already familiar with the Perl programming language, while other pieces explore some of the intricacies of the Machine Readable Cataloging (MARC) format that may be new to noncatalogers. If at any time you feel overwhelmed know that you are in good company, and keep in mind that the aim of this article is simply to show how a piece of OSS is being used in different library environments.

In short, MARC.pm is an open source Practical Extraction and Report Language (Perl) module for reading, manipulating, and writing USMARC data. As is the case with most short answers, this description glosses over several details that should be covered before going into any examples of how libraries are using MARC.pm.

First of all, for those not already acquainted, Perl is an open source programming language created by Larry Wall in 1984. Perhaps the best description of Perl is found in the documentation that comes bundled along with Perl itself:

> Perl is a high-level programming language with an eclectic heritage written by Larry Wall and a cast of thousands. It derives from the ubiquitous C programming language and to a lesser extent from sed, awk, the Unix shell, and at least a dozen other tools and languages. Perl's process, file, and text manipulation facilities make it particularly well suited for tasks involving quick prototyping, system utilities, software tools, system management tasks, database access, graphical programming, networking, and World Wide Web programming. These strengths make it especially popular with system administrators and CGI script authors, but mathematicians, geneticists, journalists, and even managers also use Perl. Maybe you should, too.[1]

Perl's facility with text processing, database access, and networking have also made it popular with librarians around the world, as more and more library services and information resources have been made available online. Perl is one of the jewels in the crown of open source software (OSS), and as such deserves the full attention of another article (or book) to detail its use in libraries.

A Perl module is a set of related functions that are packaged together into a library file that has an extension of ".pm." A module provides reusable code that extends Perl's core functionality by giving it extra powers in a new field of expertise. For example, there are Perl modules for useful tasks such as logging into and interacting with an FTP server or querying a relational database; and for esoteric jobs such as converting dates from the Julian to the Mayan calendar or interpreting genetic data. Anyone can write a Perl module; in fact hundreds of people have and made their modules publicly available in the Comprehensive Perl Archive Network, or CPAN (www.cpan.org). CPAN is a true virtual library and treasure trove of useful software, which prevents programmers around the world from constantly recreating the wheel. It is made possible in part by the professional programmers who monitor the quality of software placed there, while ensuring that the archive's namespaces remain sensible and coherent.

Most library professionals have some familiarity with the MARC data format that is used to encode bibliographic data. Catalogers in particular have detailed knowledge of the various fields that make up a MARC record, such as the 100 field for an author, the 245 field for a title, and the 650 for a subject heading. Figure 1 is a typical example of a single MARC record.

Some library professionals may not be aware that MARC data is shuttled around on the Internet every day between individual libraries and institutions like OCLC, RLIN, and the Library of Congress (LC); and that the transmission format for MARC records is quite complex. Figure 2 is the MARC record above in transmission format.

Much of this complexity paradoxically arose out of the need to make bibliographic data rigorous and compact enough to be efficiently processed by computers back in the late 1960s when the MARC data format was developed (as processing power and disk space were limited). As a result it is not a simple task for a programmer to create software that processes MARC data.

Which leads us back to MARC.pm. MARC.pm is a Perl module that extends Perl's already useful text processing abilities to make it easy for Perl users to write programs that modify and create MARC data. MARC.pm contains functions to read in USMARC data; to add, remove, and

Anne L. Highsmith (AHIGHSMI@lib-gw.tamu.edu) is Consortia Systems Coordinator at Texas A&M General Libraries, College Station. **Mark Jordan** (mjordan@sfu.ca) is Librarian/Analyst at Simon Fraser University, Burnaby, B.C., Canada. **Eileen Llona** (ellona@u.washington.edu) is International Studies Computer Services Librarian at the University of Washington, Seattle. **Peter E. Murray** (pmurray@law.uconn.edu) is Computer Services Librarian at the University of Connecticut Law School, Hartford. **Edward Summers** (ed@cheetahmail.com) is a Software Engineer with CheetahMail, New York.

modify fields; to search through your data; and to save MARC data. Perl excels as a glue language for its ability to marshal different toolsets and data sources when solving a problem. MARC.pm is designed to make Perl even more useful for librarians. Below you will find four case studies that show how MARC.pm has been used.

Editing Electronic Resource Holdings

At Texas A&M University (TAMU) the MARC.pm module has been used to edit bibliographic and holdings records that represent electronic resources. If a bibliographic record contains a URL in MARC field 856, that URL appears as a clickable link in the library's Voyager catalog. It is therefore critical that such bibliographic records have valid URLs in order to guide patrons to electronic resources. As for holdings records, the library recently decided to class nongovernment document electronic resources in the LC classification system rather than class them with a generic call number. In this latter case, it was necessary to extract the LC call number from the bibliographic record and place it in the holdings record.

TAMU's first experience with the MARC.pm module arose from a need to edit the serial bibliographic records added to the catalog when the library began participating in the Pricing Electronic Access to Knowledge (PEAK) project in the spring of 1998. At that time it was decided that separate bibliographic records would be used for the PEAK electronic serials. Bibliographic records were downloaded from OCLC and uploaded into the library's NOTIS system. Reference staff at TAMU decided that the URL in the bibliographic record should be designed to take the user to the PEAK project page at the University of Michigan rather than directly to the individual serial title in order to facilitate statistics gathering. Cataloging staff, therefore, cut and pasted the same URL into each of the PEAK serial bibliographic records.

As catalogers were adding bibliographic records for the PEAK serials to the catalog, the library webmaster was adding the same titles to a Web page that contains a complete listing of all electronic resources, including serials, to which the library subscribes. This Web page is known as the Public Access Menu (PAM). When a user clicks on the PAM, he or she is taken to a special page containing a copyright statement, any special access instructions and information, a link to the desired resource, and a link to a problem-reporting system that allows the user to report access problems with a resource. Once the user clicks on the link to the desired resource, he or she is connected to the serial title or database represented. The base URLs that support the PAM are stored in a SQL server

```
000 00897pam  2200277 a 4500
008 860317s1986    nyu     b     001 0 eng
020      $a068806499X
050 00  $aP211$b.L73 1986
100 1   $aLogan, Robert K.,$d1939-
245 14  $aThe alphabet effect :$bthe impact of the phonetic
alphabet on the development of western civilization /$cRobert
K. Logan.
250      $a1st ed.
260      $aNew York :$bMorrow,$cc1986.
300      $a272 p. ;$c22 cm.
504      $aBibliography: p. 249-265.
500      $aIncludes index.
650  0  $aAlphabet$xHistory.
650  0  $aWriting$xHistory.
650  0  $aCivilization$xHistory.
```

Figure 1. Sample MARC Record in Interpreted Format

```
00629pam  2200193 a
450000500170000000080041000170200015000580500020000731
000029000093245012600122250001200248260003200260300002
100292504003000313500002000343650002300363650002200038
6650002700408-20011105081147.0-860317s1986    nyu     b
001 0 eng  -  a068806499X-00aP211b.L73 1986-1 aLogan,
Robert K.,d1939—14aThe alphabet effect :bthe impact of the
phonetic alphabet on the development of western civilization /-
cRobert K. Logan.-  a1st ed.-  aNew York :bMorrow,cc1986.-
a272 p. ;c22 cm.-  aBibliography: p. 249-265.-  aIncludes
index.- 0aAlphabetxHistory.- 0aWritingxHistory.- 0aCivilization-
xHistory.-
```

Figure 2. Sample MARC Record in Transmission Format

database; the actual link is generated on the fly by an ASP script when the user clicks on the link. When the PAM was first started, there was no integration between the electronic resources in the library catalog and the resources listed in the PAM. For example, the catalog record for the serial title *Soil and Tillage Research* would have contained the following MARC field: 856 7_ ǀu www.umdl.umich.edu/peak/index.html. This took the user directly to the PEAK project page at the University of Michigan, whereas the PAM contained the link http://library.tamu.edu/resources/ASP/track.asp?resource=Soil+and+Tillage+Research, which takes the user to the copyright page where there is another link that leads to the page for that specific title.

By the spring of 2001 the switch from the PEAK project to Elsevier ScienceDirect had rendered the PEAK URLs in the bibliographic records invalid, so it was necessary to correct those URLs in order to direct patrons to the appropriate Web pages for the electronic serials. The bibliographic records were edited using the Perl MARC.pm module so that cataloging staff did not have to do record-by-record corrections. In addition, it was decided that it would be most beneficial to patrons to integrate the bibliographic records with the PAM by inserting a PAM-style URL in the bibliographic record. This meant that the user would be taken to the TAMU copyright page and given an opportunity to report problems with accessing each of the resources. It also increased the likelihood that the library could collect statistics on individual title use, since the PAM counts each click on each of its resources.

An SQL query run against TAMU's Voyager database identified the target bibliographic records, which were then exported using the Voyager export utility. The MARC records were edited via a Perl program that made extensive use of MARC.pm and reloaded into the catalog using the Voyager import utility. The bibliographic records for the PEAK serials required five corrections: (1) changing field 008/22 (Form of original item) to 's,' meaning electronic serial; (2) changing "PEAK Information Service" to "Elsevier ScienceDirect" in field 538; (3) deleting field 710 for "PEAK Information Service"; (4) deleting the PEAK 856 field and inserting a PAM-style 856 field; and (5) adding field 949 to log the nature and the date of the batch update via Perl.

Resetting 008/22 to 's' for electronic serial was very simple, using the unpack_008 function, changing the resulting 'Form' hash value, and restoring the 008 using the pack_008 function. Changing PEAK Information Service to Elsevier ScienceDirect in field 538 was also relatively easy, using simple string substitution techniques, as was adding field 949 to log the date and the nature of the change made in the bibliographic record. Deleting the 710 field that referred to PEAK (but only that specific 710 field) was somewhat trickier, requiring that the program loop through an array of references to 710 fields, checking each one to see if it was the PEAK 710, and eliminating the PEAK 710 if encountered. The most difficult task was extracting the title proper from the 245 field and editing it to match the form of the URLs in the PAM to construct a PAM-style URL for the 856 field. The PAM URLs had been set up with a fairly standard syntax, but one that was very different from cataloging syntax. The general rules for constructing the PAM URLs are: convert spaces to '+'; convert special characters such as ',' to their html-escape equivalents; and capitalize the first letter of each word. In addition, it was necessary to delete any characters from the 245 field that were not part of the 127-byte ASCII alphabet.

The title *Journal of Physics. A, Mathematical and general* exemplifies many of the problems encountered in this translation. This title would appear in a MARC bibliographic record as: 245 00 ‡a Journal of physics ‡n A: ‡p Mathematical and general. The PAM form of this title is: Journal+Of+Physics+A%3A+Mathematical+And +General. The greatest difficulty came in extracting the title proper, which would be tagged as 245 ‡a and its subsequent ‡n and ‡p subfields, since theoretically there can be an unlimited number of ‡n and ‡p subfields and they can appear in any order, such as, ‡a ‡n ‡p or ‡a ‡p ‡n or ‡a ‡p ‡p. The rest of the editing was accomplished using standard Perl string manipulation techniques.

All the PEAK bibliographic records were run through the Perl program. Following this, the newly constructed 856 fields were compared to a list of PAM URLs for Elsevier ScienceDirect titles to determine if they matched. If the PAM-style title devised for the 856 did not match the PAM title exactly then the user would receive a "page cannot be displayed" error if he or she clicked on the 856 in the OPAC. Once the bibliographic records with successful matches were identified, they were freshly extracted from the production database, edited as described using the Perl program, and reloaded into the database. Of the 529 bibliographic records that had contained URLs referencing the PEAK project page, it was possible to construct a correct PAM-style URL for 456. The nonmatches were due to absence of a particular title from the PAM or inconsistencies in the PAM URL. For example, the webmaster's general rule in constructing PAM URLs was to convert '&' to its html-escape equivalent of '%26amp%3B,' so that conversion was encoded into the processing for the 245 field. However, in the case of *Soil and Tillage Research*, the PAM URL is Soil+And+Tillage+ Research.

A similar procedure was followed for the second project utilizing MARC.pm, that of changing a generic call number in the MARC holdings record for an electronic resource to an LC call number. Target records were again identified by running an SQL program against TAMU's Voyager database. This second target group consisted of all holdings records that contained the generic call number "Go to URL" and their associated bibliographic records. Once identified, the bibliographic and holdings records were exported using the Voyager export utility. In general this project was simpler than the PEAK program, but did involve one interesting complication—the need to deal with two MARC record types, bibliographic and holdings. The goal of the program was to extract an LC call number from the bibliographic record if one was available and then add it to the appropriate holdings record, deleting the "Go to URL" call number from that holdings record in the process.

The first step of the program was to identify the record type using the unpack_ldr function and checking the resulting 'Type' hash value. If it was a bibliographic record, the 050 fields and 090 fields, in that order of priority, were extracted. The bibliographic record was then written to the output file and the next record checked. If the record was a holdings record and it contained 852 ‡h Go to ‡i URL, the new call number was added to the 852 ‡h or 852 ‡h ‡i, depending on which subfields were available from the bibliographic 050 or 090. Finally, a ‡m

Electronic was added at the end of the 852 field using the insertpos function. This subfield was added at the request of reference staff to differentiate these electronic resources from their print counterparts. Care must be exercised in using insertpos, since it apparently assumes that the subfields in MARC fields appear in alphabetical order and makes its insertion into a field at the first position that the inserted subfield would fall alphabetically. This worked for inserting ‡m after the final ‡h or ‡i in these records but would not work in all cases, since MARC subfields are not strictly alphabetical within fields. Once the holdings record was edited it was written to the output file, which was then uploaded into Voyager using its batch import utility. This process allowed the library to reclassify more than 750 records for electronic serials without catalogers having to perform tedious record-by-record editing.

Working with netLibrary Records

At the beginning of the 2001–2002 academic year, the University of Connecticut purchased access to approximately four thousand netLibrary titles through the local OCLC provider. The school of law, as part of the University of Connecticut campus network, has access to all of the titles. While the main campus library loaded all of the titles into their catalog, only a subset of records were appropriate for the law school catalog based on collection development practices.

Using a Perl program with the MARC.pm module, each record in the netLibrary-supplied MARC files was examined for possible inclusion in the catalog. Based on a regular expression match of the LC call number, an item was selected for loading in the local catalog. Of the four thousand MARC records, 106 matched the selection criteria. Several local edits on the records were required before loading them into the ILS.

It was decided to insert a marker field into the record so that a list of the netLibrary records could be created in the ILS for further future processing (see figure 3).

Because of the indexing rules in the law school catalog, the call number needed to be copied from the 050 to the 090 field (see figure 4).

Public Services staff also determined that the public display tag in the 856 field was too cumbersome, and recommended it be replaced by a clearer label. MARC.pm's as_string() method allows you to dump the contents of a MARC record into a variable, do transformations on it, and then read the variable back as a MARC record with the from_string() method (see figure 5).

Other edits were performed, including adding fields specific to the INNOPAC catalog for automatically setting the location and material type codes. The selected, modified netLibrary records are stored in a single MARC file,

```
# get the date in YYYY-MM-DD format
$date = sprintf(
        '%4.4d-%2.2d-%2.2d',
        (localtime())[5]+1900,
        (localtime())[4]+1,
        (localtime())[3]
);

# add the 590 field
$x->addfield({
        record          => '1',
        field           => '590',
        i1              => '0',
        i2              => '0',
        value           => [a => "netLibrary insertion $date"]
});
```

Figure 3. Inserting a Marker Field

```
($subfield_a) = $x->getvalue({
        record      => '1',
        field       => '050',
        subfield    => 'a'
});
($subfield_b) = $x->getvalue({
        record      => '1',
        field       => '050',
        subfield    => 'b'
});
$x->addfield({
        record      => '1',
        field       => '090'},
        value       => [a => $subfield_a, b => $subfield_b]
});
```

Figure 4. Copying the Call Number Field

which is subsequently loaded in the law school ILS as a batch of records.

The author of this Perl program initially had difficulties working with the MARC.pm module, finding that some of the method names were not intuitive or suggested a different function. For instance, in the first example above the getupdate() method suggests more than just a retrieval of field values.

In the great Perl tradition, MARC.pm provides more than one way to accomplish the same task. For instance, to modify a field value, one can get a copy of a field, change it, and save it to the MARC object instance using getupdate()/addfield() methods, or one can write the entire MARC instance as a string and modify it using regular expressions, as in the last example.

There are also at least two ways records can be selected to match the call number criteria. An early version of this program loaded the entire MARC file into the

```
# this will be new subfield z in the 856 field
$new_sub_z = 'Access an electronic copy of this book.';

# get the contents of the first record in our MARC.pm object as
# a string
$string = $x->[1]->as_string();

# perform a regular expression substitution to replace existing
# subfield z with out new one
$string =~ s/^(
      856\s                 # 856 fields...
      ..\s                  # with any indicators
      .*\c_z)               # everything before subfield z
      [^\c_]+               # ignore the existing subfield z
      (.*)$                 # capture what's after subfield z
/$1$new_sub_z$2/xm;         # replace subfield z with new one

# recreate record one from the transformed string
$x->[1]->from_string($string);
```

Figure 5. Transforming the 856 Field

MARC.pm object and selected records using the searchmarc() method. When using this method, the programmer must load the entire MARC file into memory using the 'increment=>-1' option. This option can use vast amounts of memory, and for that reason it was abandoned in favor of the incremental method. That method reads in one record at a time using the nextrec() method in a while loop.

The complete version of this program along with the documentation on its use can be found at www.pandc.org/peter/work/projects/parseNetLibrary.html.

Using MARC Data in a Web Application

The University of Washington Libraries prepares the Edited Works and Collections on the Middle East for the Middle East Studies Association. Catalog records for items pertaining to the topic are identified in the RLIN system and enhanced with table of contents notes in the 505 field. Perl scripts are used to export the MARC records into field-delimited text files. These files are then utilized in a Web-based program that allows searching of words with diacritics, which are heavily used in the catalog records. Search points include title and author words from both the book and chapter titles, author browsing, and publication year.

Records are downloaded from RLIN to a local disk using the pass command. The export from RLIN produces MARC formatted records, which include leader information as well as the MARC tagging. Downloading several records results in a one-line file. However, extraneous control characters often show up using RLIN's pass command. These control characters must be removed before the data can be handled via MARC.pm.

MARC.pm is used to pull out specific fields from each record. Our project uses the title (245), publication data (260), and notes fields for chapter information (505). There are anywhere from thirty to fifty records per year, with each record containing ten to thirty chapters. Thus, for a year's worth of data, there are up to fifteen hundred records representing chapters and book titles. The Edited Works list goes back to 1988, so the project will potentially include fifteen thousand records.

The following code runs through a file of downloaded records, counts them, and prints out the 245 field. A loop uses the number of records count to iterate through the file. A sample of the downloaded MARC file loops can be seen in figure 6.

A sample line of the resulting file looks like this:

> Alienation or integration of Arab youth: between family, state and street/<TAB>edited by Roel Meijer.<TAB>2000.<NEWLINE>

Obtaining specific data from the MARC records is relatively straightforward using the MARC.pm library of functions. Some understanding of arrays and loops is needed to fully utilize some of these functions (such as searchmarc), although this learning curve is not too steep. Basic books on Perl provide enough information to begin the project. The ability to pull selected fields out of the MARC record, combined with the power of regular expression searching in Perl, allows for much flexibility in creating a searchable text file or database.

Jake and Jake2marc

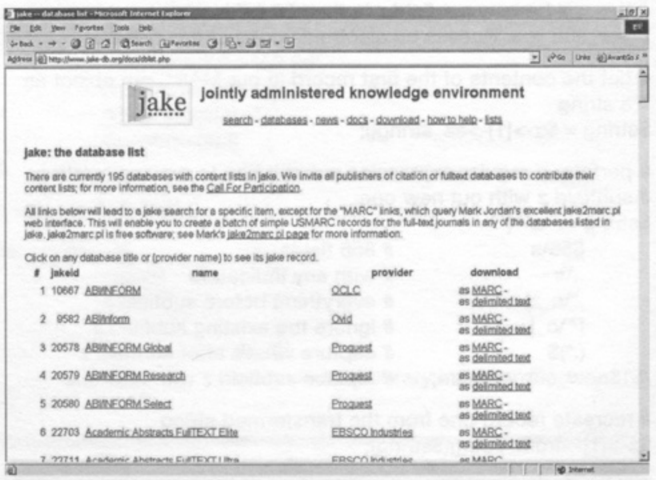
The Jointly Administered Knowledge Environment (jake) (www.jake-db.org) is an open source project that assembles information about databases of full-text electronic serials and makes that information available for reuse in a number of ways. It is used most often by reference librarians and library users to find out where a serial title is indexed, to what extent the coverage in a particular database for that title is, and to find out where full text for a serial title is available.

One of the most interesting things about jake is that it allows the information it contains to be used in many different ways (as long as those ways comply with the GNU Public License). In this respect, jake is not only open source, it is open data as well. This means users can extract information about electronic serials from jake and use it locally or repackage that information for a variety of purposes.

A Perl script that uses the MARC.pm module to create simple, standard MARC records for full-text titles

```
# create a new MARC object from file marc.dat
$x=MARC->new("marc.dat","usmarc");

# marc_count is a MARC.pm function that tells you how many
# records are in your file. The number of records is used for
# looping later on

$numrecs=$x->marc_count();

# loop through the file while printing out the selected
# fields and subfields (245  and 260 in this example),
# with a tab character in between, and a newline at the end

for ($i=0; $i<$numrecs; $i++) {
print

        $x->getvalue({
                record    => $i,
                field     => '245',
                subfield  => 'a'
        }),

        "\t",

        $x->getvalue({
                record    => $i,
                field     => '245',
                subfield  => 'c'
        }),

        "\t",

        $x->getvalue({
                record    => $i,
                field     => '260',
                subfield  => 'c
        }),

    "\n",
```

Figure 6. Creating Tab Delimited Data from MARC



jake -- database list - Microsoft Internet Explorer

Address: http://www.jake-db.org/docs/dblist.php

jointly administered knowledge environment

search - databases - news - docs - download - how to help - lists

jake: the database list

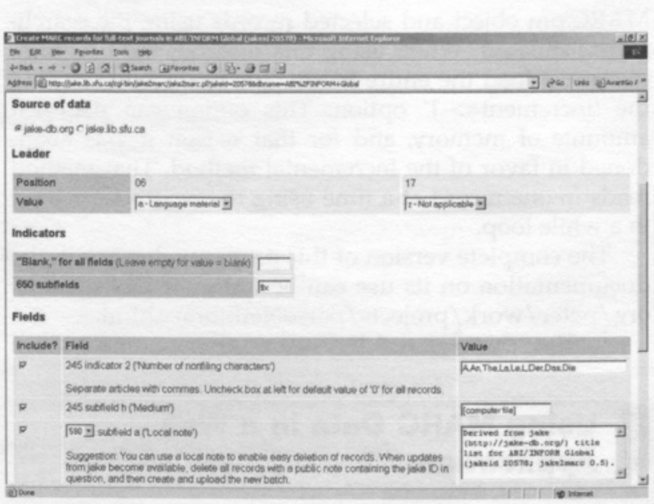
There are currently 195 databases with content lists in jake. We invite all publishers of citation or fulltext databases to contribute their content lists; for more information, see the Call For Participation.

All links below will lead to a jake search for a specific item, except for the "MARC" links, which query Mark Jordan's excellent jake2marc.pl web interface. This will enable you to create a batch of simple USMARC records for the full-text journals in any of the databases listed in jake. jake2marc.pl is free software; see Mark's jake2marc.pl page for more information.

Click on any database title or (provider name) to see its jake record.

#	jakeid	name	provider	download
1	10667	ABI/INFORM	OCLC	as MARC - as delimited text
2	9582	ABI/Inform	Ovid	as MARC - as delimited text
3	20578	ABI/INFORM Global	Proquest	as MARC - as delimited text
4	20579	ABI/INFORM Research	Proquest	as MARC - as delimited text
5	20580	ABI/INFORM Select	Proquest	as MARC - as delimited text
6	22703	Academic Abstracts FullTEXT Elite	EBSCO Industries	as MARC - as delimited text
7	22711	Academic Abstracts FullTEXT Ultra	EBSCO Industries	as MARC

Figure 7. Jake Database List



Create MARC records for full-text journals in ABI/INFORM Global (jakeid 20578) - Microsoft Internet Explorer

Address: http://jake.lib.sfu.ca/cgi-bin/jake2marc/jake2marc.pl?jakeid=20578&dbname=ABI%2FINFORM+Global

Source of data

(●) jake-db.org ( ) jake.lib.sfu.ca

Leader

Position	06	17
Value	a - Language material	z - Not applicable

Indicators

"Blank," for all fields (Leave empty for value = blank)	
650 subfields	\x

Fields

Include?	Field	Value
✓	245 indicator 2 ('Number of nonfiling characters') Separate articles with commas. Uncheck box at left for default value of '0' for all records.	A,An,The,La,Le,L,Der,Das,Die
✓	245 subfield h ('Medium')	[computer file]
✓	590 subfield a ('Local note') Suggestion: You can use a local note to enable easy deletion of records. When updates from jake become available, delete all records with a public note containing the jake ID in question, and then create and upload the new batch.	Derived from jake (http://jake-db.org/) title list for ABI/INFORM Global (jakeid 20578; jake2marc 0.5).

Figure 8. Jake2marc Options Form

described in jake is called jake2marc. Within jake's list of databases (see figure 7), users can simply click on the "download as MARC" link, which leads them to the jake2marc utility.

A number of options, such as which fields to include, what to use as MARC subfield indicators, and what local notes to include in the records are offered by jake2marc (see figure 8).

After choosing the desired options, the user submits the Web form and a few moments later is presented with a link to the MARC communications file, as well as (if the user has asked for it), a human-readable ASCII representation of the MARC records.

Taking the options indicated in the Web form, jake2marc retrieves the information it needs from jake using the Perl LWP module, which can retrieve a file from the Web using a simple HTTP GET request. This information is in roughly the same format as that retrieved if someone clicks on the "delimited text" link in the jake list of databases illustrated in figure 7. Once the delimited file describing the database is retrieved, jake2marc reads each line, which represents a serial record, creates a MARC record for the title, and then adds the record to a MARC communications file.

For each record, jake2marc creates the following MARC fields: 022 (ISSN), 050 (LC call number), 082 (Dewey call number), 245 (title statement; subfields a and h only), a local notes field (5XX; user can choose exact field number), and 650 (local subject headings). The last field, 650, is local because jake doesn't distinguish between the various types of subject headings (personal name, corporate name, geographic name) that have specific MARC field numbers. At the same time, jake2marc also creates an 856 field (electronic location and access) that holds the URL of the full-text. In addition to these title-specific fields, each MARC record also has the required record leader.

Upon its release, jake2marc generated a lot of interest on the mailing lists over which it was announced. Many catalogers, seeing the need to be able to generate simple copy cataloging for e-serials in aggregated databases, made a number of suggestions that were incorporated into incremental versions of the script.

Conclusion

Hopefully these examples have illustrated how a piece of OSS is being used in a variety of library settings. If your curiosity is piqued, more case studies, plus full documentation and the module itself are available from the MARC.pm homepage at http://marcpm.sourceforge.net. It is important to note that the Perl4lib discussion list, www.rice.edu/perl4lib, played a crucial role in the development of MARC.pm. The discussion list was started by Charles McFadden (Virginia Institute of Marine Science) in 1999 and is currently maintained by Chuck Bearden (Rice University). Perl4lib brought the developers into contact with each other, allowed them to work together remotely at the cost of an Internet connection, and provided a mechanism for connecting the project with the wider world of librarians interested in using Perl. Perl4lib is a forum for discussing all things Perl (not just MARC.pm), so if you are interested in learning more about Perl and its application in libraries, or exploring new projects with people like yourself, please visit the Web site, subscribe, and join the conversation.

Reference

1. Perlfaq1, General Questions about Perl, Revision: 1.23, May 23, 1999. Accessed Jan. 10, 2002, www.perldoc.com/perl5.6/pod/perlfaq1.html.

Authority Express

If you have been searching for an easy way to authority control your library's current cataloging, try LTI's ***Authority Express*** service.

With ***Authority Express,*** a library uses the Internet to transmit a file of newly cataloged bibliographic records to LTI (via ftp). LTI immediately processes the records through its state-of-the-art authority control system. Then, at the library's convenience, it logs into LTI's ftp server to retrieve fully authorized catalog records, along with linked LC name and subject authority records.

Authority Express

- Keeps authority control current at an affordable price
- Integrates easily into existing workflows
- Lowers cost by reducing staff time spent on catalog maintenance
- Provides 1 hour turn around for up to 1,000 catalog records

"Authority Control for the 21st Century"

LTI

LIBRARY TECHNOLOGIES, INC.

2300 Computer Avenue, Suite D-19 Willow Grove, PA 19090
(215) 830-9320 Fax: (215) 830-9422
(800) 795-9504 email: LTI@LibraryTech.Com

Visit our website at:
www.LibraryTech.Com

The EOR Toolkit: An Open Source Solution for RDF Metadata

Harry R. Wagner

Despite its unprecedented growth in popularity, the Web has failed to live up to expectations regarding its usefulness as a research tool. Technology has not kept pace with the growing number of Web sites. Libraries, the recognized experts in research and information management, have been unable to take an active lead in solving this problem. A solution is proposed, using the Resource Description Framework (RDF), an evolving metadata standard, in a collaborative open source environment that will enable libraries to take a more active role in the development of applications and services focused on improving the discovery and management of electronic resources.

In little more than a decade, the Web has become the research tool of choice for many, if not most, library patrons. The amount of information available on the Web, and the ease and speed with which it can be published and made available, offers enormous potential. It also presents a serious challenge for libraries with regards to the discovery, reliability, relevance, and accuracy of search results. Libraries need technical solutions to balance the enormous, but flawed, potential of the Web with the dependable service their patrons have come to expect.

Although libraries have long been recognized as experts in research and cataloging, the rapid pace at which information technology (IT) is changing and the limited IT budgets of most libraries have prevented them from taking a lead role in the evolution of the Web. It is increasingly difficult for libraries to become, and remain, knowledgeable of existing and emerging ITs. This is especially true for libraries with limited or no technical staff.

The discovery and information management of the growing volume of Web sites and other electronic resources is a task that will require the skills of professional librarians. Libraries need solutions that will enable them to take a more active role in the development of applications and services focused on the discovery and information management of Internet resources.

This paper presents a collaborative approach to solving these problems that relies on open standards and open source software (OSS). The library community has a long history of collaboration with, and a well-known respect for, open standards. These can be leveraged within the open source community. By pooling ideas, functional requirements, and technical skills, libraries can use their expertise to shape the Web in a way that makes it more reliable, and that benefits the library community as a whole.

This article will introduce RDF and describe how it is being used to solve many of the problems associated with Web search engines. RDF is an open standard metadata recommendation for the Web. It provides a framework for describing resources using machine-understandable semantics that enable the automated discovery, management, and exchange of metadata.

The Extensible Open RDF (EOR) Toolkit will be presented as an example of the type of OSS that is being used to create applications focused on the discovery, navigation, and management of RDF metadata. EOR is an open source project and is built entirely upon open standards. An overview of the open source distributions used to develop EOR will also be presented.

RDF

The Semantic Web Activity is a project underway at the World Wide Web Consortium (W3C). Its goal is to bring meaning, structure, and organization to the Web in a way that enables the automated discovery, understanding, and exchange of Web pages and other Internet resources.[1]

RDF is a metadata recommendation of the W3C and one of the key technologies in the Semantic Web Activity project.[2] It is an open standard XML application (though not limited to XML) and provides a framework that enables a number of important improvements in the information management of Web sites and other electronic resources. These include:

- machine-understandable semantics that will enable the automated discovery, management, and exchange of Web sites and other Internet resources;
- finer granularity and improved precision for resource discovery; and
- interoperability of different metadata vocabularies using a common syntax.

Web search engines, while much improved from a few years ago, continue to catalog only a small portion of the Internet, produce results that are sometimes irrelevant, are often inaccurate, or are missing altogether (such as, broken links).

Search engines are limited by a number of factors: the technology being used, the amount and quality of metadata available, and the sheer size of the Internet. They fall into two broad categories: crawler-based and directory-based. The main difference between the two is the method used to harvest the data.

Harry Wagner (wagnerh@oclc.org) is a Senior Consulting Systems Analyst with the OCLC Office of Research and the Dublin Core Metadata Initiative, Dublin, Ohio.

Crawler-based engines, such as Google, AltaVista, and HotBot use automated processes known as Web crawlers (sometimes referred to as spiders or bots) to "crawl" the Web searching for resources. They index the metadata and text found at each resource. The problem with this approach is that the vast majority of resources available on the Internet are written in HTML. These resources are meant to be understood by humans but not by such application processes as crawlers. This results in inaccurately indexed resources and is why search results often include inaccurate or irrelevant hits.

Directory-based search engines, such as Yahoo! and DMOZ, use manually created catalogs. These catalogs consist of Web sites that have been submitted by the site administrators or have otherwise been discovered. They have a much higher degree of accuracy and relevancy than do crawler-based engines but produce a much smaller result set because of the manual effort involved and the size of the Internet.

Sizing the Internet, a study released by Cyveillance, reported the existence of 2.1 billion unique publicly accessible Web pages.[3] This number is expected to double by early 2002. The sheer size of this number will soon dictate that resource discovery and cataloging be automated.

RDF addresses these problems by providing a framework that enables the automation of both the discovery of resources (like crawler-based engines) and the ability to intelligently index resources (like directory-based engines). RDF accomplishes this through the use of machine-understandable semantics. RDF metadata are specifically designed to be understood and exchanged by automated processes, such as user agents and search engines.

RDF improves resource discovery by providing a higher degree of precision to search results than do current search engines. For example, an RDF request for "all resources written by Stuart Weibel in 2001" would produce a result set that is both succinct and completely relevant. This is not possible with existing search engines such as Yahoo! or Google. These types of applications would produce a large number of inaccurate and irrelevant results due to their use of text-based indexing and their limited user interface. RDF syntax uses semantics that are machine-understandable and that enable even more sophisticated compound searches. Consider, as an example, the RDF request for "all resources coauthored by Stuart Weible and Eric Miller between the years 1998 and 2000 that contain the word metadata in their title." This degree of accuracy is possible because RDF provides meaning to the metadata.

Rather than attempting to define a metadata vocabulary that could be used to describe all resources, RDF builds on the established work of various resource communities by enabling the use of existing metadata vocabularies within those communities (for example, Dublin Core). This enables different communities to describe and share resources with their own unique vocabularies, using a common RDF syntax.

RDF accomplishes this with a simple syntax consisting of a resource and a number of assertions about that resource. A resource in RDF terms is anything that can be described with a uniform resource identifier (URI). A URI is a unique identifier, similar to a uniform resource locator (URL). It can be anything. It does not have to be resolvable, but it must be unique. This enables RDF to be used to describe electronic resources and anything else that can be uniquely identified.

Resources and assertions about resources form simple triples, called statements, which are the foundation of the RDF syntax. Statements are always comprised of a subject, a predicate, and an object (predicate value). The subject and the predicate must both be resources. The object can be either a resource or a literal (a character string).

RDF uses the XML namespace facility to qualify resources. This prevents element name collisions that could arise whenever multiple vocabularies are combined. The namespace facility is also used to identify the RDF schema. The RDF schema differs in both syntax and intent from XML schemas and is what enables RDF to provide meaning to the elements comprising different vocabularies.

For example, a simple assertion, using the Dublin Core vocabulary, would be: Harry Wagner is the author of the Web site http://eor.dublincore.org/eor-install.html. This can be visualized as follows in figure 1.

An in-depth discussion of RDF is beyond the scope of this paper. However, the basic principle of defining assertions using triples that are comprised of resources and literals is key to understanding RDF. This, hopefully, is enough of an explanation to allow the reader to grasp its enormous potential.

EOR

Application Overview

EOR is an open source project, developed by the OCLC Office of Research (http://oclc.org/research) and the Dublin Core Metadata Initiative (http://dublincore.org).[4] Its goal is to facilitate the rapid development of RDF applications focused on the discovery, management, integration, and navigation of metadata. EOR consists of services that can be used as applications themselves, but its real intent is to serve as a toolkit for developers writing their own metadata applications.

EOR is built on public-domain software that adheres to open standards. Eric Miller started the project in April 2000 and is still actively involved as the project lead.

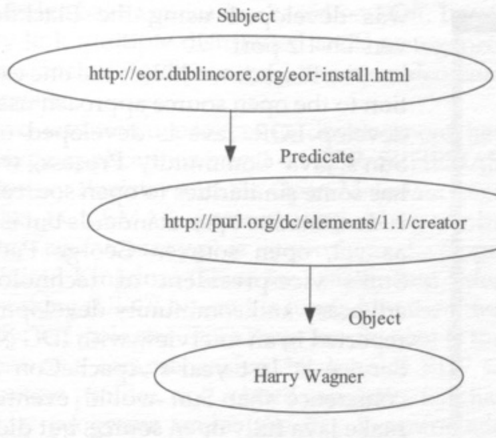

Subject
http://eor.dublincore.org/eor-install.html
Predicate
http://purl.org/dc/elements/1.1/creator
Object
Harry Wagner

Figure 1. Simple RDF Statement

Version 1.0 was publicly announced in May 2001, and the current release is version 1.01. EOR is a good example of the type of functionally rich RDF applications that are available to the library community. It is one of many open source RDF applications that are being used to build search engines and other information management applications. Dave Beckett's RDF Resource Guide (www.ilrt.bris.ac.uk/discovery/rdf/resources) is an excellent source of similar applications.

The EOR source code (available for download from the project home page) is intended to serve two purposes: to demonstrate by example functionality common to all metadata applications, and to serve as an advanced starting point for developers writing their own RDF applications. Providing this base level of functionality promotes rapid development by enabling application developers to focus their efforts on source code specific to their applications.

The project is comprised of services that provide the following functions:

- A user-authentication service that demonstrates with a simple interface how to authorize users. Authentication records are saved as RDF models.
- A validation service to validate and serialize RDF models. Models can be rendered in any of three ways: as a collection of RDF statements (triples), a serialized model, or as a graph using RDFViz (an open source RDF graphics generator).[5]
- An infuse service for saving RDF models to a persistent data store.
- A delete service to remove models from persistent storage.
- An RDF search engine that performs simple and compound searches against the data store. The search results are returned to the client session as an RDF model and then rendered into a number of formats using XSLT (see figure 2).
- A session management service that provides a support interface for managing such session objects as saved models, session attributes, cookies, and persistent models.
- An administration service for creating and deleting persistent data stores.

Each of the EOR services will run independently of the others, making them a mix-and-match grab bag for developers. For example, almost all metadata applications will provide a search interface; some might choose to provide an interface to save data to persistent storage. Others may elect to provide an interface for users to create and manage one or more personal save areas. EOR is designed to facilitate the development of each of these functionally different applications by providing source code and services that not only demonstrate that functionality, but that also serve as an advanced starting point from which to begin coding.

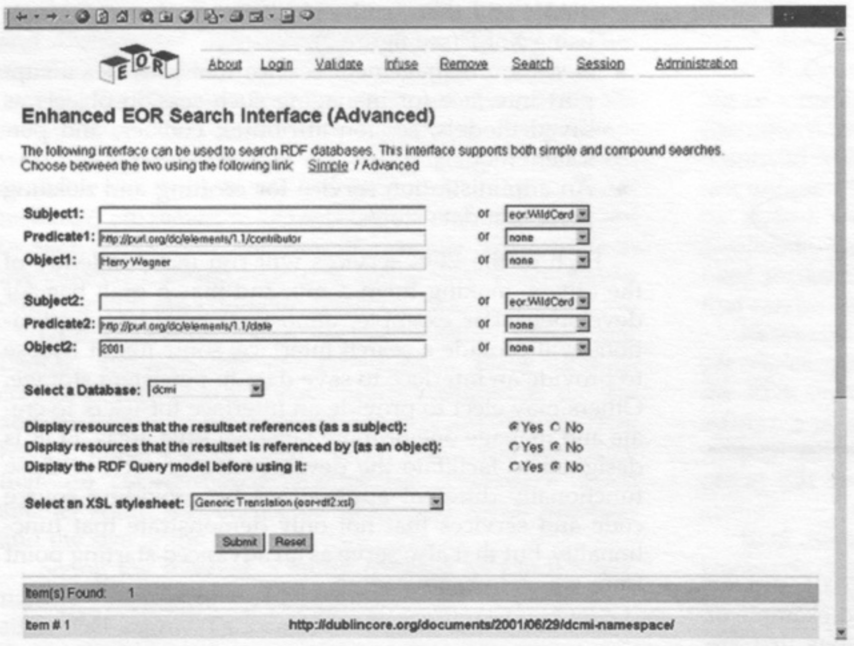
Open Source Tools Used to Develop EOR

EOR was developed using the Red Hat Linux (http://redhat.com) version 6.2 operating system. Linux is a mature, robust, multitasking operating system, initially developed by Linus Torvalds. It is the largest, and arguably the most successful, open source project to date.

The Apache HTTP Server is installed as part of most Linux installations (There is a saying within the Linux community: "Linux is like a wigwam, no gates, no windows, and an Apache inside.") and is the HTTP server used to develop EOR. Apache is without doubt the most popular HTTP server available today, and for good reason—it is feature-rich, extensible, robust, and it is open source. It is also one of the largest and most mature open source projects in existence, dating back to 1995, and is licensed by the Apache Software Foundation (http://apache.org).

EOR is a server-side Java application composed primarily of Java servlets and Java Server Pages (JSPs). EOR, like all server-side Java applications, runs in the context of an application server (sometimes referred to as a servlet engine), which is a Java-enabling extension to an HTTP server. Client requests are processed to and from the application server via HTTP.

EOR was developed with the Jakarta Tomcat (http://jakarta.apache.org) application server. Tomcat is an open source project and the official reference implementation for Java Servlets and JSPs. It was developed under the Java Community Process and was donated by Sun to the Apache Software foundation. One of the



Enhanced EOR Search Interface (Advanced)

The following interface can be used to search RDF databases. This interface supports both simple and compound searches.
Choose between the two using the following link: Simple / Advanced

Subject1:		or	eor:WildCard
Predicate1:	http://purl.org/dc/elements/1.1/contributor	or	none
Object1:	Harry Wagner	or	none
Subject2:		or	eor:WildCard
Predicate2:	http://purl.org/dc/elements/1.1/date	or	none
Object2:	2001	or	none

Select a Database: dcmi

Display resources that the resultset references (as a subject): [✓ Yes] No
Display resources that the resultset is referenced by (as an object): Yes [✓ No]
Display the RDF Query model before using it: Yes [✓ No]

Select an XSL stylesheet: Generic Translation (eor-rdf2.xsl)

Item(s) Found: 1

Item # 1 http://dublincore.org/documents/2001/06/29/dcmi-namespace/

Figure 2. EOR Search Service

benefits realized by developing EOR with Tomcat is that EOR should run under any application server that complies with the reference release.

Search results are processed by EOR's search service and returned to the client's session as an RDF model. RDF models, being an application of XML, can be rendered into HTML (or a number of other formats) using XSLT. For this, EOR uses Xalan-Java version 2 (http://xml.apache.org/xalan-j/index.html), another open source distribution from the Apache Software Foundation. The EOR source includes a number of different XSLT style sheets that can be used to render search results into different formats.

EOR uses Jakarta Ant (http://jakarta.apache.org/ant/index.html), another open source distribution from the Apache Software Foundation, for project builds. Ant is a Java-based build tool that is best described as "the Unix make utility, on steroids." Because it is Java-based, it is consistent across platforms and will run anywhere that Java runs. It also has the advantage of being able to interface natively with the Java compiler, which greatly decreases the time required for builds. It is similar to the Unix make utility in that it operates on "targets" that are specified at runtime. It uses XML syntax, and as a result, is much easier to use than the make utility.

The Blackdown Project (http://blackdown.org) describes a group of developers that were responsible for the first ports of the Java Developers Kit (JDK) to the Linux platform. Tod Matola, one of the EOR version 1.0 developers, was part of the Blackdown team and EOR was developed using the Blackdown version 1.2 port.

The Blackdown JDK is the one exception to the open source approach used to develop EOR. Java is developed using Sun's Java Community Process, which has some similarities to open source and is dedicated to open standards but is not, as yet, open source.[6] George Paolini, Sun's vice-president of technologies advocacy and community development, reported in an interview with IDG News Service at last year's ApacheCon 2000 conference that Sun would eventually make Java fully open source, but did not provide a timeline for this transition.[7] Whether or not Java will go open source, and when, is a subject of seemingly endless debate, and Sun has received some unfavorable press as a result.

Several proposals for an RDF application programming interface (API) have been presented, but none, as yet, have become standard.[8] EOR uses a sample implementation of one of the proposals, the Stanford RDF API (www-db.stanford.edu/~melnik/rdf/api.html). This API is an open source distribution and implements the Simple RDF parser and compiler (SiRPAC) (www.w3.org/RDF/Implementations/SiRPAC), which is the W3C reference implementation for RDF parsers. EOR uses the Stanford RDF API for all RDF related processing, such as parsing and serializing models, and creating resources and literals.

EOR uses a relational database management system (RDBMS) for persistent data storage and was developed using MySQL (http://mysql.com), an open source RDBMS. EOR uses a factory approach to database implementation that is designed to support all SQL92-compliant database management systems. Implementations for MySQL and PostgreSQL are currently provided.

How Is EOR Being Used?

The Dublin Core Metadata Initiative (DCMI) Web site search engine (http://dublincore.org/services/search.jsp) was built based on EOR. This is a full-text RDF search engine that provides an interface for searching the DCMI Web site. Users can specify searches across elements, or match text on any of the fifteen elements in the Dublin Core element set.

The Department of Defense, with the help of McDonald Bradley, is currently developing a virtual knowledge base application based on EOR. It will intelligently link multiple distributed and heterogeneous data sources, including databases, imagery, video, audio, and documents.

Two additional projects in development have built prototype applications based on EOR: SCHEMAS Forum Vocabulary Registry and the Open Metadata Registry.

The SCHEMAS project (http://reg.ukoln.ac.uk/registry/jsp/sforum.jsp) is currently in development by the United Kingdom Office for Library and Information Networking (UKOLN), and is funded by the Information Society Technologies (IST) Program. Its goal is the development of a comprehensive database of RDF schemas, application profiles, and related semantics that have been used by programs under the IST Program and other related European initiatives. The SCHEMAS database will be used to promote the reuse and interoperability of semantics for existing and new projects.

The Open Metadata Registry (http://wip.dublincore.org:8080/registry/registry1) has much in common with the SCHEMAS project. Its goal, like the SCHEMAS project, is the creation of a database comprised of vocabularies and related semantics belonging to various resource communities. Also like the SCHEMAS project, the Open Metadata Registry will be used to promote the discovery and reuse of semantics within existing vocabularies and the creation of new vocabularies. The real difference between the two, however, is that the Open Metadata Registry project will register vocabularies relating to the Dublin Core Metadata Initiative, whereas the SCHEMAS project will register RDF schemas and namespaces used by projects within the European Union. The Open Metadata Registry project is currently under development by the Dublin Core Metadata Initiative.

Conclusion

The fundamental issues (such as discovery, classification, cataloging) for managing information available from the Internet are not very different than those required for books, serials, and other traditional resources. The technology is different but not the underlying issues. Managing the discovery, classification, and integration of the growing volume of Web sites and other electronic resources is a task that will require the skills of professional librarians. Libraries can take a leading role in solving these problems by leveraging their collective skills and expertise in a collaborative environment to create applications and services based on RDF metadata, open standards, and open source.

RDF provides solutions that will enable a significantly higher degree of reliability, relevance, and accuracy for applications and services focused on resource discovery and management of Web sites and other Internet resources. Through its use of machine-understandable semantics, RDF enables the automated discovery, management, and exchange of metadata. It significantly improves resource discovery by enabling a finer degree of granularity and improved precision. In addition to facilitating the creation of new resource descriptions, RDF builds on the established work of various resource communities by enabling the interoperability of existing metadata vocabularies within those communities.

EOR is one of a large and growing number of open source applications that are being used to develop applications and services focused on the discovery, management, integration, and navigation of electronic resources. The number of high-quality open source distributions available in the last few years has increased dramatically. The library community can use these open source distributions, in combination with the advanced functionality provided with RDF toolkits such as EOR, to develop applications and services that enable a more effective and robust use of the Web and other electronic resources.

References and Notes

1. W3C, Semantic Web Activity. Accessed June 20, 2001, www.w3.org/2001/sw.

2. W3C, Semantic Web Activity: Resource Description Framework. Accessed June 20, 2001, www.w3.org/RDF.

3. Cyveillance, Inc., Internet Exceeds 2 Billion Pages. Accessed July 6, 2001, www.cyveillance.com/web/us/newsroom/releases/2000/2000-07-10.htm.

4. OCLC Office of Research. Dublin Core Metadata Initiative, Extensible Open RDF Toolkit. Accessed July 23, 2001, http://eor.dublincore.org.

5. Dan Brickley, Institute for Learning and Research Technology, RDFViz. Accessed July 23, 2001, http://rdfviz.org.

6. Sun Microsystems, Community Development of Java Technology Specifications. Accessed July 2, 2001, http://jcp.org.

7. Laura Rohde, Network World, Inc., IDG News Service, Sun Says Java Moving towards Full Open Source. Accessed July 16, 2001, www.nwfusion.com/news/2000/1024javasource.html.

8. Peter Hannappel, Summary of Recent Discussions about an Application Programming Interface for RDF. Accessed July 9, 2001, http://nestroy.wi-inf.uni-essen.de/rdf/sum_rdf_api.

SPI

An ISO 9002 certified company

Content Conversion Services

Experts in SGML, HTML, XML, Electronic and Imaging Conversions

PacificData
CONVERSION CORP.

207 E. Forest Drive, Box 396 • Pocono Pines, PA 18350
Phone: (570) 646-5302 • FAX: (570) 646-5317
email: ihanney@epix.net • www.spitech.com/pdcc

Open Source, Open Standards

Karen Coyle

When people speak of open source software they are referring to computer code—programs that run. But code is only the final step in the information technology process. Prior to writing code the information technology professional must do analysis to determine the nature of the problem to be solved and the best way to solve it. When software projects fail, the failure is more often than not attributable to shortcomings in the planning and analysis phase rather than in the coding itself. Open source software provides some particular challenges for planning since the code itself will be worked on by different programmers and will evolve over time. The success of an open source project will clearly depend on the clarity of the shared vision of the goals of the software and some strong definitions of basic functions and how they will work. This all-important work of defining often takes place through standards and the development of standards that everyone can use has become a movement in itself: open standards.

Open standards are publicly available standards that anyone can incorporate into their software. An example from the library environment is the MARC record standard. The original documentation for the MARC record was published by the American National Standards Institute.[1] The most common use of the standard, that of the MARC21 records that libraries adhere to, is also published and available for use. No one owns the MARC record format; there are no fees for its use and no restrictions on who can use it in their products. Any software developer who wishes to write for library systems therefore has access to a vital part of the system needs: the basic data structure that libraries use today.

This may seem so obvious that its importance is hard to grasp. In fact, the library world has probably made more use of open standards than practically any other industry. Let's face it, "open" is practically our middle name. Examples from the non-open world of proprietary software might help us understand the importance of our preference for open standards, and the examples are not hard to find: Microsoft Windows versus the Macintosh operating system; VHS versus Betamax; Nintendo versus Sega. In each case you have unique products that are inherently incompatible. As a matter of fact, this incompatibility is purposeful and actually enhanced by the companies in question as part of their market strategy. If you need to compete, then openness is a disadvantage. If you need to cooperate, then openness is the way to go.

Goals of Open Standards

Open standards can serve multiple needs. The most common one is the need for interoperability. Interoperability refers to communication between systems or system parts. In the highly networked world of the twenty-first century, the ability for computer systems to exchange data in order to carry out basic functions is absolutely vital since most systems operate in a vast and varied digital community. Our library systems communicate electronically with sources of bibliographic records, book vendors, and users. They also now interconnect themselves with networked information resources outside of the library and deliver these through library-maintained interfaces. Much of this communication is through open-standard interfaces, such as Z39.50, Electronic Data Interchange (EDI), and hypertext transport protocol (HTTP).[2] These standards operate at the point where system boundaries touch; they determine the rules of the digital membrane but do not determine how systems handle data up to that point of permeability. Internally, few systems store bibliographic data in the format prescribed by ANSI Z39.2, the basis for the MARC record. But they are able to transform the data into that format for communication with other systems.

Another purpose of open standards is to create the framework for a community. In many ways this is the prime reason for many library standards. The use of common cataloging rules does not so much allow libraries to intercommunicate as it does create a certain look and feel and a commonality between libraries that is an aid to users. It allows users to move between libraries without having to learn a whole new process for finding materials, and it makes it possible for the library profession to train librarians and hire from among a pool of candidates. The cataloging rules, published and readily available to anyone with the desire and patience to learn them, contributed to the rise of professional (rather than artisan) librarianship. Creating the rules brought members of the library community together to ponder not only the vagaries of title pages but also to confront some basic philosophical issues about the organization of knowledge.

Today, in a world where many activities are performed through computer programs, open standards can be promulgated as a way to encourage decentralized development. Much of the work of the World Wide Web Consortium (W3C) falls into this area. The W3C is a membership-sponsored standards body that creates new standards for the

Karen Coyle (www.kcoyle.net) is a Systems Developer at the California Digital Library, Oakland.

Web. These standards can be used by anyone writing software for the Web. What is critical about many of these standards is that they set the foundation for entirely new Web functions; functions that will only work if many different people develop their part of the software that is needed.

This is rather hard to describe but should become clear with an example. I'll use the recent development of the Platform for Privacy Preferences (P3P).[3] P3P is a set of rules that allows Web sites to describe their privacy practices in a standard way. It would also allow Web users to express their "privacy preferences" using the same standard vocabulary. P3P does not specify how this will be implemented on the Web; the development of actual software will be left to the rather amorphous Web community. For P3P to be part of the Web, it will be necessary for Web site owners to incorporate P3P into their sites, and for Web browsers to create a user interface to the function. But for P3P to be successful, it needs to be recognized by all major browsers (Internet Explorer, Netscape, and AOL), and it must be used by a large number of Web sites. Since many companies and institutions make use of software like FrontPage or Cold Fusion to develop their large and complex Web sites, tools for building P3P will need to be included in these packages. By specifying a standard for privacy preferences, the W3C is attempting to set in motion a very decentralized software development project that will need to be undertaken by a wide variety of players.

Sort of Open versus Really Open

Although we speak about open standards, some are more open than others. This is because there are a variety of aspects to open standards, and standards that call themselves open do not always adhere to all of these.

Open standards are:

- standards that anyone can use to develop software or functions;
- standards in which anyone can participate in their development and modification; and
- standards that anyone can obtain without a significant price barrier.

The best example of standards that meet all of those criteria are those created by the Internet Engineering Task Force (IETF). The IETF dates back to pre-Internet days, when it was a group of engineers working on the first developments that eventually became the basis for that network. These engineers developed a way of chronicling and communicating their technical ideas through a series of documents called Requests for Comments or RFCs.[4] The first RFCs were almost in the form of notices ("OK, I'm going to send packets with a 5-byte header, let me know if you can read them"), but as time went on the RFCs became well-thought-out standards that had been developed by groups of volunteers. Anyone can comment on the RFC, either to point out errors or to make suggestions. Even after the technical decisions in the RFC are accepted and implemented, the RFC remains an RFC. Some RFCs improve or comment on previous ones, as technology changes or as better ideas arise.

The functioning of the IETF is like a lesson in democracy: one person or a group of people sees the need for a new or modified function for the Internet; they draft a proposal which is placed on the Internet for anyone to read and comment on; if the proposed function meets a need and is successfully tested with an actual program, it becomes part of Internet use. The IETF is open to anyone who wishes to participate. That last statement needs qualification, however: participation in the IETF requires a high level of technical knowledge and a considerable amount of a person's time. Those who make up the various IETF committees are a self-selected technocracy. And while the philosophy of the IETF is one of engineering "purity," today's committees invariably have members who represent technology companies that often have a particular bias toward their own products. Still, there is no other standards organization that is as open as the IETF, and there is still considerable input from the academic and research communities.

This can be compared to the W3C, the standards organization formed to develop and promulgate standards for the Web. Participation in the W3C is limited to members—predominantly technology companies—who pay between $5,000 and $50,000 per year to belong to the group. Compared to the IETF, this group is lacking the academic and research engineers who bring a financially neutral viewpoint to the discussions. There are also almost no members who might represent a public interest viewpoint. This latter is significant because the W3C does not limit itself to standards of engineering; there is an effort called Technology and Society (within which P3P was developed) that develops standards for functions like content filtering and privacy.

There are a number of other standards bodies, such as the National Information Standards Organization (NISO), the International Standards Organization (ISO), and the American National Standards Institute (ANSI). These organizations have members who participate directly in the development of standards. The standards, once developed, are not only open for use, but some of them are actually mandatory within certain industries. Obtaining the actual text of the standards is, however, another question.

Standards-making is an expensive enterprise and standards bodies have traditionally made money on the sale of the printed form of their standards. Since many companies and organizations would be required to adhere to the standard, this provided a kind of guaranteed audience for the standards documents, many of which carried rather hefty price tags. The W3C, having arisen from the Internet community (and with the example of the IETF preceding it) makes its standards available for open access on the Web. In comparison, the document from ISO describing the Universal Character Set which all modern computing is moving toward is priced at about one hundred dollars. Although it isn't a huge price if viewed in light of the research and development budget of a company, it does make it difficult for small organizations, nonprofits, schools and libraries, and individuals to make active use of the standards. Responding to these needs and to the move toward greater openness in the standards area, in 2000 NISO became the only national standards organization making its standards available over the Internet for free. There is some risk because this removes a significant revenue stream from the organization. The gain is that the organization should be even more successful in its primary mission, which is that of providing standards for widespread use.

Open Standards and Libraries

The first of the library technology standards was the decision at the first annual ALA meeting in September of 1877 to standardize the catalog card at 7.5 x 12.5 cm.[5] While this was intended to make mass production of cards possible (and by analogy more standardized production of card cabinets as well), the advantages of an open standard manifested themselves when in 1898 the Library of Congress (LC) began its printed card service. This was possible only because libraries in the United States were using the same sized card and thus filing into cabinets that held cards of that size. We can consider the LC card service the technological predecessor of the MARC record service of the latter half of the twentieth century. The card-size standard was its key to interoperability.

The next technological standard of great interest was the computerization of those same cards through the MARC record standard. Prior to the development of what we now think of as MARC, a group of librarians led by Henriette Avram of LC developed a machine-readable record format standard for bibliographic data, ANSI Z39.2. This standard made use of other national standards, such as the ASCII character set. Although at the time only LC had the capability of producing the records (and the motivation to do so), this is arguably the most significant technological development of modern librarianship. By establishing an open standard for machine-readable records, LC created the basis for the computerization of library catalogs. That wasn't the intention in 1965 when Z39.2 was proposed, however. LC was focused on automating its card production services and creating a print-on-demand card service. Like Dewey's desire to reduce the cost of card-stock production, the LC standard, because it was open, was available to be used in ways that its creators had not yet imagined.

Few library open standards have been as successful as the MARC standard. Since 1965 arguably the most widely used standard is Z39.50, the protocol for information retrieval from remote databases. Z39.50 takes advantage of the existence of searchable bibliographic databases in library automation systems and the networking provided through the Internet. The protocol had a somewhat slow beginning, partly due to its complexity, but today the functionality is included in most library system packages and there are even open source versions of the software.

Other standards have been less successful. One example is the Common Command Language (CCL), Z39.58. CCL is a standard set of commands for searching in online catalogs that was developed by NISO in 1992. When the standard came up for its five-year NISO review the organization's members allowed the standard to lapse. Although some systems claim to use a common command language, these generally do not use the standard commands defined in the NISO standard. So how did a standard become not a standard after all?

The reason for creating a common command language was not unlike one of the original motivations behind a standardized set of cataloging rules: the uniformity between libraries makes it easier for users to move from library to library. A common command language is especially important in current times because users may be using a number of library systems almost simultaneously over the Internet. Why would such a useful standard fail? There are a number of reasons why standards might not be adopted. One of the obvious ones in terms of the CCL standard is the fact that the technology that the standard responded to, the command-line interface to library databases, was eclipsed by a new technology, the Web browser.

Although some command-line searching remains, it is not the main user interface. Another reason for the lack of adoption of the CCL is something that gives standards development a tricky aspect: people seem less likely to accept standards that affect the content aspects of their computer systems. Successful standards tend to define background functions, and leave a great deal of flexibility for system developers in terms of presentation. For example, the protocols that control the Internet e-mail function do not dictate how e-mail will be presented to the user. Everything from the command line Pine e-mail software to the almost user-obsequious Microsoft Outlook product make use of the same e-mail protocols. Yet another reason is that standardizing the command line gains you very little where the underlying indexes of the system are not themselves standardized. The command line is merely the interface to a much more complex set of decisions about what fields feed into what indexes, and about how the data in those language-based fields is treated for the purpose of searching.

The lesson here is that not all aspects of systems are ideal candidates for normalization. Whether rational or whimsical, system developers clearly express a need to have a certain amount of freedom. Standards need to facilitate functionality without suppressing the creativity of system developers or their ability to meet the needs of their particular target audience. Standards work best in the underlying technology layers and less well the closer one gets to the actual user.

Some library standards currently in development might fit this bill. For example, the NISO Circulation Interchange Protocol (NCIP) standard for interlibrary loan (ILL) is intended to facilitate interoperability between library systems for ILL transactions.[6] ILL is an obvious area where communication between diverse systems is needed for automation of the function.

Libraries don't live by library standards alone, however. Increasingly our library systems are interacting with the wider world of technology, delivering library services over public networks. We use mainstream standards such as the Internet protocols developed by the IETF, the Web protocols of the W3C, the character sets defined internationally by the ISO. Library representatives were heavily involved in the latter effort, having already participated in the development of a similar standard known as Unicode. However, there is virtually no library participation in organizations such as the IETF or W3C, even though the standards developed by these organizations are vital to our operations. Not only are libraries missing from the standards groups, so also are schools and nonprofit organizations, which are kept out not only by the membership fees but also by the labor requirements for active participation: the need to dedicate a significant amount of time of a highly skilled technical worker to the standards process.

While it is unlikely that individual libraries would be able to be active in a standards organization, we now have a possible model for greater library participation: in 2000, the American Library Association (ALA) joined the Open eBook Forum (OEBF), an industry group working on e-book standards. By leveraging the strength of ALA's membership it has been possible to spread the burden of participation while at the same time provide a visible library presence for the standards process.

Conclusion

The Internet has given us an entirely new model for the cooperative development of highly complex systems and subsequently of the standards that allow those systems to work. Although the Internet has not lived up to some of the Utopian promises of its early days, it still allows a low entry barrier for active participation; so low in fact that individuals can create their own Web sites right on the same network beside those of major companies. We might not be able to give all of the credit to the IETF and its example of open standards, but it is clear that open standards are an essential element in the success of the Internet and its widespread use. Continuing the open standards tradition will be essential for its continued success.

References and Notes

1. National Information Standards Organization (U.S.), *Information Interchange Format* (Bethesda, Md.: NISO Pr., 1994.) National Information Standards Series ANSI/NISO Z39.2-1994.
2. National Information Standards Organization, *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification* (Bethesda, Md.: NISO Pr., 1995.)
3. Full documentation on P3P is available at www.w3.org/P3P. Accessed Oct. 2, 2001.
4. There are a number of sites that house searchable copies of the IETF RFCs. The official IETF RFC site is www.ietf.org/rfc.html. Accessed Oct. 2, 2001.
5. Wayne A. Wiegand, *Irrepressible Reformer: A Biography of Melvil Dewey* (Chicago: ALA, 1996): 53–54.
6. NISO Circulation Interchange Protocol is a draft standard, available for review and testing. Accessed Oct. 2, 2001, www.niso.org/committees/committee_at.html.

Related URLs

American National Standards Institute
  www.ansi.org
Internet Engineering Task Force
  www.ietf.org
International Organization for Standardization
  www.iso.org
Library of Congress MARC Standards Office
  lcweb.loc.gov/marc
National Information Standards Organization
  www.niso.org
Open eBook Forum
  www.openebook.org
World Wide Web Consortium
  www.w3.org

ProCite 5.0

2141 Palomar Airport Rd., Ste. 350
Carlsbad, CA 92009
1-800-722-1227
info@isiresearchsoft.com

Price: Full retail version $395.95, students $109.95, single-use upgrade $99.95.

System requirements: Microsoft Windows 95/98/NT4/2000/XP. Word processor compatibility: Microsoft Word for Windows 7, 97, 2000; Corel WordPerfect 7, 8, 9.

Macintosh version system requirements: PowerPC/Mac with Macintosh operating system or compatible (120MHz recommended); Apple MacOS System 7.5.5 or later.

Reference Manager 9.5

2141 Palomar Airport Rd., Ste. 350
Carlsbad, CA 92009
1-800-722-1227
info@isiresearchsoft.com

Price: Full retail version $395.95, students $109.95, single-use upgrade $99.95

System requirements: Microsoft Windows 98/ME/XP/NT4/2000. Word processor compatibility: Microsoft Word for Windows; XP, 2000, 97 and 7; Corel WordPerfect 8, 2000, 2002.

Anyone who has ever published a scholarly book or paper knows that the most tedious aspect is generating the bibliography. Nor is any task associated with this so prone to avoidable error. ISI ResearchSoft has made this task a little less tedious with its suite of bibliographic management products. Bibliographic management software eliminates the need for writing citations on file cards for later use. Citations can be entered into the database manually; they often can be imported from a database on the Internet. Once the citations are in the bibliographic reference manager file, they can be retrieved by a variety of means and used for articles, books, and any other publication requiring a bibliography or citations. The reference manager software automatically reformats the citations according to any one of the hundreds of predefined output styles, which vary from journal to journal.

Two of these products, Reference Manager and ProCite, are reviewed here. Endnote was reviewed in the December 2000 issue of *ITAL*, but an updated version was recently released, and the differences between versions are addressed in these pages as well.

ProCite and Reference Manager can be used apart from a word processing application to search the Z39.50 databases, to search for references within a bibliographic file, or to generate a bibliography. Conversely, they have the capability to integrate themselves into Microsoft Word or WordPerfect via their Cite While You Write (CWYW) function. This allows the author to format citations and generate a bibliography with a few simple commands.

Installation on a Windows NT 4.0 machine running Microsoft Office 2000 was quick and straightforward. ProCite and Reference Manager both integrated seamlessly as add-ins with Word during installation and should do so with WordPerfect. Installation results in a new toolbar as well as a new pull-down menu item for Word. When you install either of these products, the installation software determines which versions of Word or WordPerfect are available and installs the CWYW function automatically. However, if Word or WordPerfect is installed after installing either of these products, the CWYW setup must be run for the function to appear in the word processor's Tools menu.

ProCite has a Macintosh version available, while Reference Manager only exists in a Windows version. The look and feel of both products is nearly identical. In fact, the toolbar is identical and only slight differences exist in the pull down menus. Of course, this means that they have many functions in common as well. Each can search Internet databases, organize references, and format bibliographies. A bibliographic file created by ProCite, Reference Manager, or EndNote can be easily converted to a form usable by any other of those three applications.

A primary function of a bibliographic manager is to create, manipulate, and format bibliographic references. This is a formidable task when the number of bibliographic references in a scholarly work can number in the hundreds or even thousands. Of the many capabilities of this type of application, managing references is one of the more mature functions. A particularly useful function is the ability to reformat citations to meet the demands of a particular journal. Both Reference Manager and ProCite allow more than six hundred output styles, such as MLA, Chicago, and APA. Another important function is to facilitate searching of Z39.50-compatible Internet databases for relevant bibliographic references, although this function is less sophisticated than the bibliographic management ability.

In terms of basic functionality, both products allow users to search Z39.50 databases, organize references, and format bibliographies. Each application allows unlimited references. Reference Manager allows thirty-five fields in each reference while the ProCite allows forty-five. In terms of reference type (monograph, journal, working paper), ProCite is the clear leader in this area with fifty predefined types and the ability to add more. Reference Manager has thirty-nine predefined types, but limited ability to add types.

Reference Manager and ProCite allow the user to enter citations manually, but their real convenience is the

Christian Poehlmann (poehlmann.1@nd.edu) is Manager of the Business Information Center at Mendoza College of Business, University of Notre Dame, Indiana.

ability to import citations from databases. These two products will, via appropriate filters, import the results of these searches into the bibliographic reference manager. ProCite and Reference Manager have filters for more than three hundred Internet databases and online library catalogs. Additionally, the user can create filters manually. The search/import functions are not as mature as the formatting functions and require a significantly greater expertise and a steeper learning curve. However, a handful of selected data services allow direct export using a plug-in from ISI ResearchSoft. The Export Plug-in is a free download from the ISI Web site that installs a utility and import filter required to export references from ISI, Sea Change, and BioMetNet Web-based products into ProCite and Reference Manager. Most of these data services are within the ISI universe of products and exhibit a strong bias toward scientific literature.

Reference Manager is a feature-rich writer's tool for researchers, offering three research tools in one: a reference searcher, a database manager, and a bibliography builder. The reference searcher allows the user to retrieve references from existing Reference Manager databases, or from an Internet Z39.50 database using Boolean logic. Once references from an Internet database have been retrieved, they can be imported into an existing Reference Manager bibliographic file or saved to a new file. The database manager function allows the user to copy references within or between databases and to copy data from one reference to another. It also allows the user to edit, add, or delete authors and journal titles in the database. The bibliography builder function allows the user to insert citations into documents using the CWYW add-in. Once citations are inserted, Reference Manager can then generate a bibliography. Workgroups may benefit from multiuser read and write access capabilities of the network edition. Reference Manager is available as a true network application with multiple read/write access to the same database—down to the field level. This capability allows multiple users to make changes simultaneously to the bibliographic database.

As the same company publishes all three products, enough cross functionality exists between the products that it no longer makes any real sense to continue the product line with three separate and distinct titles. EndNote and ProCite differences are minimal. EndNote has multilingual spell check capability, while ProCite lacks any spell check function. On the other hand, ProCite has reference grouping and advanced searching capabilities that do not exist in EndNote. Each exists in both a Macintosh and Windows version, and each lacks true network capability. Reference Manager, on the other hand, lacks a Macintosh version but has true networking functionality.

The searching function is of limited utility. Most of the proprietary databases searchable via the Z39.50 standard require password authentication rather than the standard academic practice of using IP authentication. This is not an issue, however, with most online catalogs and free databases.

New in Reference Manager

As this review was being completed, ISI released version 10 of Reference Manager. Although this release was not reviewed, ISI announced the incorporation of several new features. Among the more interesting is CWYW with Instant Formatting. This feature, available to users of Microsoft Word, creates the bibliography as the author types. In previous versions, the bibliography was generated after the citations were entered into the document. Then, using the Generate Bibliography command on the Tools menu, the users clicked on a dialog box. Each of the citations entered was converted to the in-text citation format in the desired style (MLA, APA). The list containing cited references was then appended to the document. With this new feature, the only time the Generate Bibliography command is used is to change the selected output style or to modify layout options. Preformatted bibliographic styles now number more than seven hundred.

Other features include the enhanced ability for users of Microsoft Word to collaborate with colleagues using a traveling library. This library is created automatically as references are cited. When the document is sent to colleagues for editing, the library, containing all of the necessary bibliographic data, goes with it. Another new functionality is the ability to store links to other resources in each reference. Some examples of related resources might be full-text articles, image files on the Internet, or a network hard drive.

Users from institutions that subscribe to ISI Web of Science can now create searches with Reference Manager. Users with institutional access to Web of Science may purchase key reference data via ISI eSource on a pay-per-view basis.

New in ProCite

Just as with Reference Manager, ProCite gives Microsoft Word and Corel WordPerfect users CWYW ability. Some enhancements to CWYW are: the option to see the full record, including the abstract, when selecting from multiple matches; the ability to click sort on column headings to locate a reference when selecting from multiple matches; and the ability to select from any of the last ten citation searches to bring up quickly the results of a previous search.

The ability to define and modify citation formats in each workform to allow accurate footnote formatting is also a new feature. (Workforms are fill-in-the-blank forms used to help organize the information in each record.) When a source is cited more than once in a document, many style guides require an alternate format for the second and subsequent in-text citations. ProCite now allows users to define an alternate format to handle this issue. Rounding out the list of new formatting features is the ability to preview the current workform definition while creating or editing output styles.

Language enhancements also appear in this newest version. Languages vary in their rules for sorting characters, and ProCite has the capability to sort according to the current language setting. Czech, Polish, and Russian (Cyrillic) have been added to the international sort options.

What's New in EndNote?

There are several new features in EndNote 5, including enhanced word processor support for Microsoft Word 97/2000. It is now possible to locate and insert citations without leaving Microsoft Word. Users can instantly format citations as they are cited.

According to a press release dated February 1, 2002, users of OCLC FirstSearch can now also export information to EndNote:

> Using ISI ResearchSoft's Direct Export feature, FirstSearch users may now export one or more bibliographic records directly from OCLC FirstSearch into EndNote software. This will facilitate the creation of a bibliography of material accessed in FirstSearch databases during the review of search results.[1]

However, the most interesting EndNote development is the traveling library that follows the document for easy collaboration. EndNote 5 automatically creates a reference list as you cite references in Microsoft Word, containing all of the bibliographic information for all of your citations. When sending this document to colleagues for editing, you will have provided them with all the necessary information to add, delete, and reformat citations.

Conclusion

Authors in scientific fields will benefit most of all from these three products, but even those who produce scholarly works in the arts and humanities will find a bibliographic manager to be a near necessity.

Reference

1. OCLC, FirstSearch News, February 1, 2002, Online Computer Library Center, Inc. Accessed February 22, 2002, www.oclc.org/firstsearch/en/fsnews.htm#new.

Index to Advertisers

ALA Editions	37	LITA	18, cover 3
Axonix	cover 4	OCLC	cover 2
infoUSA	11	Pacific Data Conversion	32
Libary Technologies	26	University of Washington	25

LITA Preconferences

at the
2002 ALA Annual Conference
in Atlanta

What is XML? How can XML be Used in Libraries?

Friday, June 14, 2002 - 9:00am - 5:00pm

For information about this preconference
visit www.lita.org/ac2002/xml.html

Working with Open Source Software: a hands-on workshop

Friday, June 14, 2002 - 9:00am - 5:00pm

Due to the "hands-on" nature of this preconference, space is limited. Contact the LITA Office to ensure space is available. For instructions see www.lita.org/ac2002/opensource.html

The Library and Information Technology Association (LITA)
is a division of the American Library Association

NETWORK CDs... WITHOUT CDs

SuperCD

AFFORDABLE AND FAST

Network your CD-ROMs today, but don't settle for yesterday's technology. Forget those big, slow and expensive CD towers. Using one high-speed CD drive and TurboCaching, SuperCD mirrors CD disks to a large hard drive. Super CD eliminates the need for multiple CD drives. It's super fast and super affordable, easy to use and easy to install.

Thousands of satisfied users appreciate:

- **Lowest cost per CD.**
- **Networks up to 512 CD titles.**
- **Serves 1,000 users.**
- **Fastest access times plus 66X transfer rates.**
- **Compact, fits on a shelf.**
- **Supports NT, Netware & Mac, simultaneously.**
- **Web Browser accessible.**
- **DVD compatible.**
- **Unburdens your LAN.**

BEST OF LAN TIMES 1996 SOFTWARE

Buy with confidence. Axonix backs its products with a FULL ONE YEAR WARRANTY. Plus, if you're not completely satisfied, you can RETURN THE PRODUCT WITHIN 30 DAYS for a complete refund. Axonix has a 18-year history and we understand the value of customer satisfaction.

Check our website for more SuperCD information.

For a free catalog, contact:
Axonix Corporation
844 South 200 East, Salt Lake City, UT 84111
800-866-9797 or sales@axonix.com

EDUCATION DISCOUNTS ARE AVAILABLE.

Tomorrow's Library TODAY

1.800 866.9797
www.axonix.com

AXONIX®
Share and Save