Open Source Software and the Library Community

Written by
Kevin S. Clarke

A Master's paper submitted to the faculty of
the School of Information and Library Science of
the University of North Carolina at Chapel Hill in
partial fulfillment of the requirements for
the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

May, 2000

Approved by:

___________________________________,

Advisor

Kevin S. Clarke. Open Source Software and the Library Community. A Master's paper for the M.S. in I.S. degree. May, 2000. 67 pages. Advisor: Gregory B. Newby.

This paper examines whether the library community should support the development of open source software, software whose licensing allows its source code, the part of a computer program that is readable by humans, to be redistributed and/or modified without restriction or charge. Included is an investigation into the present state of library-specific open source software and a discussion on the ideological and practical strengths and weaknesses of open source software in general.

In addition, this paper explores the similarities between the “hacker culture” of the programming world and the traditional “gift culture” of the library community. Problems implementing open source software solutions are discussed, as are the strengths and weaknesses of library-specific open source software. In the final analysis, the support of open source software is seen as a positive step for the library community as it builds on the community’s strengths while minimizing its weaknesses.

<u>Headings:</u>

Free computer software

Information retrieval – social aspects

Community collaboration

Online catalogs

Table of Contents:

Libraries and librarians have, for many years, been using, creating, and sharing software, scripts, and batch files. Recently, however, with the growing popularity of the Open Source movement, some librarians have started to promote open source software, software whose licensing allows it to be redistributed and/or modified without restriction or charge, as a solution to the problems associated with traditional closed-source proprietary library software. They argue that the goals and methods of the Open Source movement are similar to those of the traditional library community and that, by bridging the gap between the two, each can benefit from the other's experience and knowledge. Should libraries promote Open Source as a style of software development? Answering a question like this requires that we look at the purpose of librarianship and examine how libraries' present and future needs might be met by encouraging the development of open source software.

Introduction

First, what is open source software? There are actually several, sometimes competing, interpretations of what Open Source, or Free Software, is. Perhaps the best, though certainly not the most definitive, way to describe open source software is to say that it is software created by communities of programmers who share their source code, that part of a program that is readable by humans, with anyone who might find the program, or a variation of the program, useful. Some programmers prefer to call this shared software "Open Source" while others prefer to call it "Free Software". Free, here, refers to one's freedom to use rather than freedom from price (Stallman, 1998a). The dispute over which name is best is, in part, a difference of opinion on how software should be licensed. To understand the reasons behind this disagreement, a brief history of code sharing is necessary.

A Brief History of Shared Software

Richard Stallman, founder of the GNU Project, the first intentionally formed software sharing community, observes that software sharing, in its various forms, started with the development of computers. He says, “[Sharing code] is as old as computers, just as [the] sharing of recipes is as old as cooking” (Stallman, 1998). Eric Raymond, author of *The Cathedral and the Bazaar* and co-founder of the Open Source Initiative, agrees but says, for practical purposes, that the birth of the Open Source, Free Software, culture we that know today can be conveniently dated to 1961 (Raymond, 1999). It grew out of MIT’s Artificial Intelligence Laboratory where Stallman worked throughout the 1970s and early 1980s.

This sharing of code, that started in isolated research centers around the United States, reached a critical mass with the development of the Department of Defense’s ARPAnet, a transcontinental, high-speed computer network built as an experiment in digital communications (Raymond, 1999). The ARPAnet linked defense contractors, research communities, and universities together in what came to be the Internet’s predecessor. It was over these high-speed networks that the world’s first programming community grew. With the growth of this new community, programmers developed their own culture, a culture that produced a variety of cultural artifacts, the Jargon File being, perhaps, the most well known (Raymond, 2000).

As the development of new computer technologies advanced, the older models became obsolete. These newer models had their own operating systems but, unlike the old systems, their operating systems were proprietary; users were required to sign a non-disclosure agreement before they were allowed to receive and use the system’s software. Many, like Richard Stallman, did not approve of the new rules. He says, “This meant that the first step in using a computer was to promise not to help your neighbor. A cooperating community was forbidden. The rule made by the owners of proprietary software was, ‘If you share with your neighbor, you are a pirate. If you want any changes, beg us to make them’" (Stallman, 1998).

Believing he could not, in good conscience, create software whose use was restricted, Stallman resigned from MIT and started the first organization intentionally formed to create free software, the GNU Project. While the goal of the GNU Project was to write a freely available UNIX clone, they are perhaps better known for the non-kernel elements of HURD, their developmental UNIX-like system. Many of these elements, combined with the Linux kernel, now constitute the GNU/Linux operating system, a system that has, recently, garnered a good deal of press, as has open source software in general. This enormous growth in popularity has been documented by Eric Rauch who, by searching Lexis/Nexis for key terms related to the open source movement, plotted references to the GNU/Linux system, and to open source software in general, in an easy to read graph (Rauch, 1998).

In 1985, to promote the software that the GNU Project was producing, Stallman created the Free Software Foundation (FSF), a tax-exempt charity for the development and promotion of free software (Stallman, 1999). Around this time, Stallman also wrote the GNU General Public License (GNU GPL) (Stallman, 1991). The GNU GPL is a license that insures that no programmer or organizational entity can subvert the right of a GPLd program to be shared with others in the programming community. The GNU license has been called a “copyleft” license because it seeks to guarantee the right of the programming community to use another programmer’s freely shared source code; this is unlike a copyright, which only guarantees the author’s right to restrict the use of his or her program (Stallman, 1999a).

In the late 1980s, the Free Software movement gathered momentum. Many programmers, distressed by the lack of a decent propriety compiler, chose the GNU Project’s free compiler, GCC, as their development tool of choice. Some of these programmers, interested in learning why such a useful tool was being given away for free, read the philosophical texts published by Richard Stallman and the Free Software Foundation. Michael Tiemann, one of these programmers, saw a blueprint in Stallman’s work for what he believed could be a successful business model. He says:

> Suffice it to say that on the surface, [Stallman’s Manifesto] read like a socialist polemic, but I saw something different. I saw a business plan in disguise. The basic idea was simple: Open Source would unify the efforts of programmers around the world, and companies that provided commercial services (customizations, enhancements, bug fixes, support) based on that software could capitalize on the economies of scale and broad appeal of this new kind of software (Tiemann, 1999).

Tiemann realized, in short, that the Free Software development model could, as a business model, accomplish corporate goals in a way that most proprietary software companies’ business models could not.

In 1989, Michael Tiemann joined with David Henkel-Wallace and John Gilmore to co-found Cygnus Solutions, the world’s first Open Source software company. The creation and initial success of Cygnus Solutions proved that free software could be profitable, and that a commercial venture could be sustained and even succeed by capitalizing on the Open Source development model. While Stallman had argued for the development of free software from an ethical/political perspective, Tiemann realized that free software could also succeed for “competitive, market-driven reasons” (Tiemann, 1999). Was he right?

In 1998, almost a decade after its creation, Cygnus Solutions was still the largest and most successful Open Source company in the world (McHugh, 1998). It was this long term commercial success that led Red Hat, the most popular U.S. distributor of the GNU/Linux operating system, to acquire Cygnus Solutions in November of 1999 for an estimated $624 million, using money it had raised from its own incredibly successful initial public offering (LinuxDevices.Com, 1999). It is worthwhile to note that for the three years before its acquisition, Cygnus Solutions consistently ranked on Software Magazine's list of the top 500 software companies in the world (Software Magazine, 1999).

Throughout the 1980s and 1990s, many programmers worked on open source projects based on the Berkeley Standard Distribution of UNIX (BSD). Unfortunately for the community as a whole, these programmers often had disagreements that, because of the centralized nature of their development model, resulted in divisions in the BSD community (Raymond, 1998b). Some of these disagreements led to the FreeBSD, OpenBSD, and NetBSD distributions, among others. Legal troubles also stunted the growth of the BSD movement. AT&T, which owned the rights to an earlier proprietary UNIX operating system, sued the BSD movement, just as it was gaining momentum, claiming that BSD programmers were using code from the older proprietary UNIX operating system that AT&T owned (Sullivan, 1999). The suit was eventually dismissed, but it slowed the acceptance of BSD within the software industry. Despite these impediments, the BSD movement is, today, still successful. Many active Internet sites use FreeBSD (Vaughan-Nichols, 1999); in addition, BSD was selected by Apple Computer as the foundation for its new operating system, Mac OS X (NetBSD Community, 1999).

According to Eric Raymond, many of the early BSD programmers were pragmatists; they were not motivated by the ideology of Free Software advocates like Richard Stallman. For them, the GPL and other open source licenses were tools to provide alternatives to the software industry's giants not a way to eliminate commercial software altogether (Raymond, 1998b). In fact, many of these programmers still prefer the more permissive open source licenses that allow for the proprietary use of open source code; one example of this type of license is the BSD License (Open Source Initiative, 1999). The BSD developers represent the middle ground between the ideological goals of the Free Software Foundation and the proprietary desires of much of the business world. Since the GPL has been referred to as a "copyleft" license, developers in the BSD camp have been suggested that the BSD license is a "copycenter" license (Lehey, 1999).

Then in 1992 something happened that would bring the Open Source movement into the public's eye, Linus Torvalds began working on what would become the Linux kernel (Torvalds, 1999). A student at the University of Helsinki, Torvalds wanted a UNIX-like operating system for his 386 computer. Unable to find one, he decided to build his own, based on a developmental UNIX-like system called Minix. Eventually, all the old Minix code was removed and the GNU Project's non-kernel elements were added; the system became what we know today as the GNU/Linux operating system (though many people just refer to it as Linux). What started as one college student's hobby, now has an estimated 7 to 10 million users worldwide and is the operating system chosen for more Internet servers than any other available, including Microsoft's (DiBona, Ockman, & Stone, 1999).

While Torvalds did initiate the development of the Linux kernel, it is important to note that he did not do the majority of the coding. Nor is it the kernel's source code alone, though that is impressive, that distinguishes Linux as an Open Source marvel. Eric Raymond, author of *The Cathedral and the Bazaar* and co-founder of the Open Source Initiative, (1999d) says:

> In fact, I think Linus's cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model. When I expressed this opinion in his presence once, he smiled and quietly repeated something he has often said: "I'm basically a very lazy person who likes to get credit for things other people actually do."

What makes the GNU/Linux system so successful, Raymond believes, is the decentralized development model that Torvalds employed to create it. This was unlike any other modern free software project up to this time; earlier projects, like the GNU Project and BSD distributions, had centralized hubs of activity. Linus Torvalds opened up the development of the Linux kernel to anyone who had access to a computer, an Internet connection, and the skills to contribute.

Torvalds agrees with Raymond that this made an important difference and helped distinguish Linux from many earlier free software projects. It is, he says, Linux's decentralized, collaborative environment that makes the GNU/Linux system notable:

> 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 (Torvalds, 1999).

What is it about this decentralized collaborative environment that attracts so many talented programmers? It might be that Torvalds' model attracts the pragmatic programmers in addition to the idealist ones; in *Homesteading the Noosphere*, Eric Raymond (1998b) says:

> Not until the Linux explosion of early 1993-1994 did pragmatism find a real power base. Although Linus Torvalds never made a point of opposing [Richard Stallman], he set an example by looking benignly on the growth of a commercial Linux industry, by publicly endorsing the use of high-quality commercial software for specific tasks, and by gently deriding the more purist and fanatical elements in the culture.

Some programmers might also be attracted to the radical openness of the community that Torvalds created. Raymond (1999a) says, “Linus Torvalds's style of development [is to] release early and often, delegate everything you can, be open to the point of promiscuity...”

This unique development style intrigued programmers like Eric Raymond, so, he decided to study it. He decided the best way to understand Torvald’s development method was to put it into practice with a project of his own, Fetchmail (Raymond, 1999h). The results of this experiment are documented in Raymond’s work, *The Cathedral and the Bazaar.* While Torvalds can be credited with popularizing a model of development that has revolutionized the software industry, Raymond should be credited as the movement’s most effective proponent. His article, *The Cathedral and the Bazaar*, caught the attention of Netscape and convinced them to release their popular World Web Web browser under an Open Source license. Raymond and other open source developers, recognizing that this was a great opportunity for the Open Source movement, co-founded the Open Source Initiative (OSI) to promote Open Source software (Raymond, 1999c).

The term “Open Source” was not actually used in the free software community before February of 1998. It was created in part to clarify that the word ‘free’ in free software referred to a ‘freedom to use’ rather than ‘freedom from price’ (Stallman, 1998a), but more importantly, in Raymond’s mind at least, to “sell the idea [of Open Source] strictly on the same pragmatic, business-case grounds that motivated Netscape” (Raymond, 1999c). Many free software programmers accepted the term, and it is the term the mainstream media prefers, in part, because of the OSI’s publicity efforts, but not all developers think that abandoning the ‘free software’ label is a good idea. Richard Stallman, for instance, has expressed concern about the de-emphasis of the ethical reasons for writing free software (Stallman, 1999b). The emphasis on getting commercial interests to accept free software, he believes, is wrong headed in that it allows the software industry to capitalize on the programming community, often without returning anything to the community from which it takes. For the most part, though, the term Open Source seems to have become the standard term for shared software.

It is hard to read anything about the software industry now without hearing about the wild success of Open Source software or the latest Linux IPO to cash in on the Open Source craze. Mainstream web sites like Time Magazine have started permanent web sites to chronicle the latest Open Source news (Time Magazine, 2000). Despite the commercial success of companies like Cygnus Solutions and Red Hat and the mainstream recognition of open source software, most in the Open Source movement acknowledge that the future of open source software rests in hands of the open source community of developers. Michael Tiemann, co-founder of Cygnus Solutions and now CTO at Red Hat, Inc., says, “The people who think money is the engine are the people who are wrong. Money is just the gas, and the engine is the open source development community. And the quality of the people who are in that community determines how much horse power this movement's going to have” (Dougherty & Sims, 2000).

The Purpose of Librarianship

In 1931, Shiyali Ramamrita Ranganathan, one of the founders of modern librarianship, conceived of five simple, yet profound, laws of library science: books are for use, every reader his book, every book its reader, save the time of the reader, and a library is a growing organism (*Encyclopedia of Library and Information Science*, 1968, p. 67). These five laws, though somewhat dated by their language, are just as applicable to the practice of library and information science today as they were in 1931. To put Ranganathan’s laws into modern terms, one might say: resources are for use, access should be available to every person, resources satisfy user needs, user satisfaction is the goal, and libraries must evolve.

Library resources are for use. This is true regardless of whether the resource is stored in the library or whether access to the resource is provided through the library. Unfortunately, in the past, library use has been restricted to particular groups of users. In early Roman times, and as far back as the Librarian King Asurbanipal’s time, libraries were open to the public, but later, starting in Medieval times, libraries were regarded as institutions for preserving and storing scrolls and books, rather than for promoting their use. A. K. Mukherjee says:

> In Medieval times … books were actually chained to the shelves … During the period of the Renaissance and Reformation, transition started from the Middle Ages to the modern world. The chains were removed at first, although access was limited to chosen few. The use of payment became the vogue. As a further step, books were

> made free to all, but only for use in the premises of the library. Then came lending to favoured few: then to all who paid the fee: and at last, lending free to all (1966, p. 32).

While some libraries still serve to archive and preserve materials, the idea that books, or other media, are merely storage containers for society’s wisdom has passed from favor in the profession. Today, library resources are collected, organized and made available, primarily, for use by the patron (Morgan, 1999a).

With this historical change have come changes in the cultural assumptions of librarians and library administrators. Some in the profession suggest that librarianship, as a result of these changes, has become a “gift culture” (Morgan, 2000a). Eric Raymond in his *The Cathedral and the Bazaar* explains what a gift culture is; he says, “Gift cultures are adaptations not to scarcity but to abundance. They arise in populations that do not have significant material scarcity problems with survival goods” (Raymond, 1998). Gift cultures can be found in many civilizations around the world. One such example of a gift culture practice is the Native American Potlatch. The Potlatch is “a ceremonial feast of the American Indians of the northwest coast marked by the host’s lavish distributions of gifts or sometimes destruction of property to demonstrate wealth and generosity with the expectation of eventual reciprocation” (Merriam-Webster, 2000).

In a private email, on February 10, 2000, Eric Morgan, a librarian at North Carolina State University, suggested that librarians share this “gift culture” with other groups like the Open Source programming community. He said, “Libraries have an abundance of data and information … Librarians do not exchange this data and information for money; you don’t have your credit card ready as you leave the door. Libraries do not accept checks. Instead the exchange is much less tangible."[1] Morgan makes sure not to confuse data and information with knowledge, but points out that the measure of a library's worth is often directly related to how much it gives away. He says:

> The items of value are information and information services exchanged for a perception of worth--a rating valuing the services rendered. This perception of worth, a highly intangible and difficult thing to measure, is something that the user of library services 'pays', not to libraries and librarians, but to administrators and decision-makers. Ultimately these payments manifest themselves as tax dollars or other administrative support. As the perception of worth decreases so do tax dollars and support (Morgan, 1999).

If we accept that librarianship is a gift culture, Ranganathan's first law makes even more sense, but what about Ranganathan's other laws?

Access should be available for every person. In a paper on the nature and future of the academic community, Peter Lyman (1997) says, "Scholarly publications are consumed within a gift culture institution called the library, a subsidized public good within which knowledge appears to the reader as a free good." That access should be free or, more accurately, subsidized is perhaps the greatest assumption of modern librarianship. One need not look much farther than the largest professional organization for librarians, American Library Association (ALA), to realize that "equity of access" is considered by most librarians to be a central purpose of modern librarianship. An ALA advocacy pack from their web site says, "Libraries have historically served as the nation's great equalizers of knowledge, providing access to information regardless of the ability to pay" (American Library Association, 2000). In *Libraries, An American Value*, the American Library Association (1999) says, "Libraries in America are cornerstones

[1] Morgan, Eric (eric_morgan@ncsu.edu). (2000, February 9). Subject: Gift cultures, librarianship, and open source software development. Email from

of the communities they serve. Free access to the books, ideas, resources, and information in America's libraries is imperative for education, employment, enjoyment, and self-government.”

Free access to library resources, like free access to software, has often been confused with a freedom from cost. This is not how the word free, in this context, should be understood. Richard Stallman (2000) says, “Free software is a matter of liberty, not price … Free software refers to the users' freedom to run, copy, distribute, study, change and improve the software.” Likewise, the *Code of Ethics of the American Library Association* says, “In a political system grounded in an informed citizenry, we are members of a profession explicitly committed to intellectual freedom and the freedom of access to information. We have a special obligation to ensure the free flow of information and ideas to present and future generations” (American Library Association, 1995).

That access should be available to every person or, as Ranganathan would say, “Every reader his book” is a fundamental assumption of modern librarianship. This does not mean, however, that there are no costs associated with providing this freedom. Libraries are supported by academic institutions, receive donations, and/or are supplemented by tax dollars. Free software, on the other hand, is often made available on a web site for no charge other than the cost of an Internet connection and, at the same time, sold on a compact disc for a profit; free software can also be given, freely, from one person to another. Richard Stallman (2000) says, “You should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being

author to rivers@clatsop.cc.or.us, arhyno@server.uwindsor.ca, and kevin_clarke@unc.edu.

free to do these things means (among other things) that you do not have to ask or pay for permission.”

Unrestricted access to every user is an ideal that libraries must often defend. In the quest to provide, “Every reader his book,” libraries are often criticized for allowing objectionable material to reach those who might not be able to make their own judgements about its validity and worth. To this, the ALA, through its *Library Bill of Rights*, has responded that free access, through the library, is necessary for all members of society. Access is necessary, librarians believe, for those who possess the ability to distinguish between a variety of information sources and for those who need to develop the skills necessary to distinguish between these questionable sources. An ALA interpretation (1997) of the *Library Bill of Rights* says:

> Those libraries with a mission that includes service to minors should make available to them a full range of information necessary to become thinking adults and the informed electorate envisioned in the Constitution. The opportunity to participate responsibly in the electronic arena is also vital for nurturing the information literacy skills demanded by the Information Age. Only parents and legal guardians have the right and responsibility to restrict their children's and only their own children's access to any electronic resource.

Both librarians, represented by the American Library Association, and programmers in the free software community, represented by the Free Software Foundation, share the assumption that unrestricted freedom is a good thing.

Ranganathan’s third law, “Every book its reader,” or as I have restated, “Resources satisfy user needs,” approaches his second law from the opposite angle. For every item in the library there is a user who would find that item useful or enjoyable; the resource, it is assumed, was purchased for good reason. Mukherjee says, “[Ranganathan’s] third law as such, runs the gamut of library operations devised mechanically or through human agencies, fulfilling the objective of ultimate utilization of every reading material, acquired with a purpose” (1966, p. 34). In short, Ranganathan’s third law says that librarians should connect a resource to a library user’s needs. This can be done through a reference interview, by cataloging the item in such a way as to make it apparent to the patron who needs it, or by creating a library catalog whose interface facilitates the end user’s search.

The goal of connecting resources to patron needs is not new. Ernest Richardson, in 1927, said, “For these [that know the information they want but not the resource in which it can be found] we furnish reference books, bibliographic cataloging, classification on the shelves and reference service to show the book that they ought to want and then proceed as before to serve it, or to beg, buy, or borrow it for them” (1975, p. 60-61). Connecting a resource to the patron’s needs is not always an easy task. In many cases, patrons are in an anomalous state of knowledge (Belkin, Oddy & Brooks, 1982). They cannot successfully articulate their needs for information in a verbal form (Taylor, 1968). In addition to this, many do not have predefined search criteria (Hildreth, 1982). Librarians, as information professionals, create library mechanisms (the formalized reference interview, the online catalog, a classification system) intended to reveal, in the most efficient way, those resources that the library owns, or to which it provides access, that might satisfy the nebulous information needs of the end user.

These mechanisms bridge the gap between the fragmented understanding of a single individual and the vast store of knowledge available through libraries. Richardson says:

> As an image of the whole of things, the knowledge of any one man is poor, fragmentary, and as a rule, rather confused heap of jumbled impressions and cognitions. The classification of these ideas organizes the confused ideas into the representation of an orderly universe, and gives unity, coherence, and integrity to knowledge and personality … Any one man by himself would never get far toward the complete picture, but happily knowledge is cooperative. It grows by each man producing something, recording it in books, and gathering the books into libraries … [The task of the librarian] is to help [patrons], and to help anyone is to cooperate with him carrying out his own plan or wishes, to help him help himself (1975, p. 57-59).

The image of librarians collaborating with each other and with the library's patrons to satisfy the patrons' current information needs is not that different from the image of open source programmers collaborating to solve their own needs. Both are working together to solve problems that exist because a single individual's understanding is, by its nature, incomplete.

Eric Raymond (1999g) touches on the necessity for collaboration in his work, *The Cathedral and the Bazaar* when he says, "While coding remains an essentially solitary activity, the really great hacks come from harnessing the attention and brainpower of entire communities." Information seeking, too, is essentially a solitary activity because its goal is the satisfaction of the seeker's individual information needs. Librarians, working with information seekers, bring to the process the collected knowledge of the library community; often, they consult other reference librarians or the mechanisms that have been constructed to represent the accumulated wisdom of the community (the catalog, classification scheme, etc.)

Fortunately, for librarians and the library's patrons, the information seeking process is iterative. The experience of the librarian, in attempting to discern and satisfy the library patron's information needs, often leads to ways to improve the library communities' retrieval mechanisms. For instance, extra access points to bibliographic records are often added once there is the recognition, often gained through a reference interview, that such a link would be useful. This "scratch your own itch" approach to maintaining the usefulness of the library's mechanisms can also be found in the programming community. Raymond (1999g) says, "It is truly written: the best hacks start out as personal solutions to the author's everyday problems, and spread because the problem turns out to be typical for a large class of users." If we think of the library's mechanisms for information retrieval as being similar to the shared code of an open source program, understanding the importance of Ranganathan's third law, "Every book its reader," makes sense.

We can also understand, in this context, why Raymond (1999f) says, "Given enough eyeballs, all bugs are shallow." Finding the right reader for a book is similar to finding the right programmer for a bug in the code. Open source programmers open up the source code to the community, and librarians open up the library's collection, and unique organization of knowledge, to the library community, both hoping to match something they possess with someone who could use or benefit from its use. In the ideal situation, after using the source code of an open source project, or the mechanisms of the library, an individual also contributes something back to the system in such a way as to improve the whole for the next person.

Ranganathan's fourth law, as restated by the author, "User satisfaction is the goal," speaks to a central focus of librarianship and open source programming. Both librarianship and software programming are goal oriented. While both have developed philosophical and theoretical foundations through which the practice of each is informed, programming and librarianship are, at their core, pragmatic. Programmers seek to solve the problem at hand by writing code that accomplishes a certain task or collection of tasks. Librarians create a unique library product and related service that can be used by the patron to solve his or her immediate information needs. This focus on pragmatic tasks has sometimes been bemoaned in the library community (Danton, 1975), but for the most part developing a theoretical foundation for library science has taken a back seat to satisfying library users’ needs.

Ranganathan’s original formulation of his fourth law, “Save the time of the reader,” also captures the need for efficiency that is desirable when working with such a complex process. Mukherjee says, “The processes of service generally generate procrustination in libraries intent on thoughtless formalities, which can be easily gotten rid of …” (1966, p. 34). The modern library does as much as possible to reduce impediments to the user’s ability to find the desired resource in the least amount of time. There are many things, internal and external to the library, that contribute to a potentially poor user experience. For instance, in this age of information overload, many librarians are turning to technology as a way to save the time of the patron.

Recently, the administrator of one open source library portal’s email discussion list took an informal survey of why librarians were interested in the project; “reducing information overload” was perceived by participating librarians as being the biggest advantage to creating and sharing an open source personalized library interface. Supporting answers from individual librarians included, “To help our users circumvent wading through ever-growing lists of electronic resources” and “A well designed personal interface keeps the main contact local but enables access to the whole -- but mainly it helps the student better and more useful[ly] use [his or her] time" (Morgan, 2000b).

Librarians recognize that saving the time of the reader is not something that can be done once and then forgotten. It is an ongoing process. As technology has evolved, many in the information community have witnessed a growing information glut. This abundance of information has led to an increased need for personalization, but many librarians and information professionals ask how much personalization is too much (how many information portals do we need before the portal, itself, becomes an impediment?) (Morgan, 2000b). Also of concern to librarians is the library's ability to provide support for patrons who have no computer experience.

This constant change is not just a result of new technologies being introduced and accepted. Ranganathan's fifth law, "a library is a growing organism," spoke to these issues back in 1931, before the modern Information Age. Still, the particular changes that shape the libraries and librarians of today are often shaped by technological evolution. Indiana University's School of Library and Information Science (1998) says, "Libraries are changing. Once passive storehouses, they have become active agents of change and early adopters of new information and communication technologies." The school's home page suggests that evolving libraries need librarians that can adapt. "Librarians are changing, too. The 'informatization' of society has generated unprecedented demand in all walks of life for specialists who will function as information resource managers and as guides, interpreters, integrators, brokers…" (Indiana University, 1998).

Assuming that the librarians can adapt, can the patrons? Will more time need to be spent helping patrons? What does this mean for the development of future library information systems? Some patrons fear that the influx of technology will create a much larger workload for public service librarians. Cate Corcoran (1997) says, ”Unfortunately, the same technology that cuts costs and relieves librarians of work behind the scenes increases it for the public -- and for the librarians at the front desk who have to help the public figure out how to use the technology. The unhappy result: People are simply not finding the information they seek.” While library software can automate many of the day to day tasks, librarians need to consider whether the adoption and promotion of library software will actually save the time of the patron.

On the other hand, librarians are here to serve patron needs and if the patrons want more electronic resources, as they seem to (Corcoran, 1997), librarians must adapt to these demands or face the possibility that their role as information brokers might be replaced by others who can. So, how do libraries implement technology in a way that maximizes its strengths without losing that which librarians already do well? As one would expect, these problems are not new to the library community. Libraries deal with these issues individually (San Francisco Public Library, 1998) and through their national organizations (Association for Research Libraries and Online Computer Library Center, 1999). This paper suggests that these issues must be approached from philosophical and practical perspectives. The author believes that the traditional philosophy of librarianship informs the practical application of these new approaches to the age old problems of librarianship, as stated by Ranganathan’s five laws, in a unique way that is necessary for the continued growth of the profession.

Open Source Software and Librarianship

The tension between the practical application of library science and the practice of relying on librarianship’s theoretical foundation to inform our daily activities remains a central component of modern librarianship. Manfred Kochen says, “It will not suffice for information professionals to have only specialized skills. They must also be humanistically enlightened generalists…” (1983, p. 281). On the other hand, this type of generality is, in itself, a specialty, says Jesse Shera (1970). In thinking about the decisions that libraries and librarians make, some have suggested that libraries do not just deliver society’s knowledge without also changing in the process. They believe the library is an iterative system. Michael Buckland (1983) points this out when he says that libraries are “open systems”; they are affected and, as a result, affect the society of which they are a part.

What type of software and software development planning best suits librarians, libraries, and the patrons? Are open systems best developed using open software, software that allows for the ongoing collaboration between community members, or are libraries better off relying on third party vendors to provide closed systems, as they have in recent years? The answer often depends on whether one believes that libraries are primarily a business or a service institution. Historically, most would agree, libraries have been subsidized service institutions, but there has always been the tension of finding that balance between cost effectiveness and serving the needs of the patrons. Ernest Richardson argued, in 1927, that the primary purpose of librarianship was to serve the library’s community; he says, “Nevertheless, the main thing about librarianship, even as a business, is not business but learning of knowledge” (1975, p. 52). Since that time, and probably before, the same concern has been echoed in a variety of forums (Kilpatrick, 1997).

Recently, a group of eighty academic library administrators met to discuss librarianship and the future of the traditional library. The product of this meeting was a paper they called the Keystone Principles. The Keystone Principles, named after the location of the meeting place, are “a set of principles and action items to guide academic libraries’ efforts and establish a foundation for joint future-oriented action based on traditional academic library values” (the Association for Research Libraries, ARL, and the Online Computer Library Center, OCLC, 1999). These eighty administrators, representing the Association for Research Libraries and the Online Computer Library Center, for the first time in a national library organization’s history, stated that the development of open source software was desirable. They say, “Libraries will create interoperability in the systems they develop and create open source software for the access, dissemination, and management of information” (ARL and OCLC, 1999).

What is more significant, perhaps, than the mention of open source software, in particular, is that the principles and the actions that the administrators recommended echo many of the same concerns heard from programmers in the open source community. This suggests that these two communities are, in fact, similar and that a collaboration between the two would be beneficial in addressing their shared concerns. This paper suggests that using open source software, as the library administrators have recommended, is the best way to accomplish the library’s goals and insure that the library remains a relevant institution in the future.

The Keystone Principles begin by stating what the American Library Association and most librarians have insisted on for years, that access to information is a public good. It says, “Scholarly and government information is created at the expense of public and/or academic institutions. Therefore, there is a public interest in the availability of this information” (ARL and OCLC, 1999). The type of information to which the Keystone Principles refer is the unique organization that libraries create and modify to make stored information accessible. While much of the information contained in an academic library is created in the academic or public realm, the library itself also modifies and creates information in such a way that distinguishes it from other information service institutions. While the library does create a unique product (its organization of information, accessible through the public catalog or browsing the stacks), most librarians still think of the library as primarily a service institution.

In contrast, closed source software companies think of themselves as manufacturers. They create a product, just like libraries, but view the sale of that product as their primary purpose; the service they provide, helping libraries use the product, is secondary. This is unlike the open source software community. Eric Raymond says, “As long as the software industry continues to misperceive itself as a manufacturing industry, instead of a service industry, reliability is going to be awful. But that shift is not going to happen until source is open. That's the difference between closed and open source” (Leonard, 1998). The open source software community, though providing a product, is a service community. The emphasis is not in getting a marketable product out the door, but in an ongoing relationship with a product and its users. This is similar to the library community. Librarians are not trying to just get an answer for the patron at the reference desk, but are teaching the patron how to use the library so that the patrons next interaction with the library will be more informed; it is an ongoing process.

The second user-centered goal of the Keystone Principles is for libraries to create bias-free systems. Most libraries today are using third party systems for the library’s day to day applications. The Keystone Principles say, “To date, these systems have been created largely outside of academe and most certainly outside libraries, thus they exist without the benefit of the expertise gained by librarians in how information is used and the academic and societal values librarians bring to the enterprise” (ARL and OCLC, 1997). Open source software created by librarians would have the advantage of incorporating the experience of librarians, something that most commercial third party vendors do not do. This is not to say that these companies do not hire librarians, but that, if they do, they are a small minority and probably not actually doing the programming. This is unlike the library specific, open source software discussed later in this paper; those programs are created for librarians, by librarians.

Whether a commercial entity can create a bias-free system has also been questioned. The Keystone Principles say, “Further, in the online environment commercial access services are distorting search results for profit without defining how these results are obtained and organized” (ARL and OCLC, 1999). This is possible in a closed-source system but not in an open one. Raymond says, “The answer is massive, independent peer review. You wouldn’t trust a scientific journal paper that hadn’t been peer reviewed, you wouldn’t trust a major civil engineering design that hadn’t been independently peer reviewed, and you can’t trust software that hasn’t been peer reviewed either.” (Leonard, 1998). Peer review is familiar to academic and public librarians. Why shouldn’t it be a part of library software as well?

As one would expect, the Keystone Principles are a mixture of ideological principles and practical suggestions for libraries of the present and the future. The principles suggest that libraries take a more direct role in managing their role as the information hub of the modern educational community, that libraries be proactive; they also consider that libraries' resources are usually limited. The standards of collaboration that define the open source community appeal to those in the field of librarianship, but so do the cost considerations of sharing the development costs of building library specific software. In addition to the ideological reasons to support open source software, libraries are finding plenty of economic and practical reasons to consider open sourcing their library software development.

As documented by Microsoft engineer/employee Vinod Valloppillil, open source software has many practical advantages over closed-source proprietary software (Raymond, 1998a). Open source software can be developed faster, can be debugged faster, has long term credibility, has a better release rate, and has an advanced programming interface evangelization/documentation method that is unrivaled among closed-source software distributors. Open source software also has some weaknesses that libraries should consider before choosing to promote or implement an open source solution. According to Valloppillil (Raymond, 1998a), open source software faces organizational incredibility, has process issues, and has high management costs.

That open source software can be developed and debugged faster than closed source proprietary software is for most part an example of its economy of scale. Even the largest proprietary software company, Microsoft, can not employee as many people as an open source project. Since open source projects are Internet based, they draw coders from around the world, each with his or her own approach to specific coding problems. This is, in part, what Michael Tiemann realized after reading Stallman’s free software manifesto. On this, Eric Raymond quotes Jeff Dutky, when he says, “Debugging is parallelizable.” Raymond (1998a) also notes that Dutky “observes that although debugging requires debuggers to communicate with some coordinating developer, it doesn't require significant coordination between debuggers. Thus it doesn't fall prey to the same quadratic complexity and management costs that make adding developers problematic.”

Valloppillil also notes that with open source software there is the possibility for “impulse debugging”. Valloppillil says:

> An extension to parallel debugging that I'll add to Raymond's hypothesis is ‘impulsive debugging’. In the case of the Linux OS, implicit to the act of installing the OS is the act of installing the debugging/development environment. Consequently, it's highly likely that if a particular user/developer comes across a bug in another individual's component -- and especially if that bug is ‘shallow’ -- that user can very quickly patch the code and, via internet collaboration technologies, propagate that patch very quickly back to the code maintainer. Put another way, OSS processes have a very low entry barrier to the debugging process due to the common development/debugging methodology derived from the GNU tools (Raymond, 1998a).

An example of “impulse debugging” in a library specific, open source project is the quick fix that the author of this paper wrote for the Open Source Digital Library System’s MARC record loader. The initial implementation of the loader stored an integer as the MARC record’s unique identification number. The problem with the program was that most libraries now use a mix of letters and numbers for unique identifiers. On trying to load some sample MARC records, the author noticed the problem, wrote a quick and simple fix, and submitted it to the OSDLS email discussion list (Clarke, 1999). It was a very shallow bug that could be “fixed” in just a few lines code, but the case proves the potential for “impulse debugging” in open source projects.

Valloppillil has also observes that open source software (OSS) has the “perfect” API evangelization/documentation method. He says, “OSS's API evangelization/developer education is basically providing the developer with the underlying code. Whereas evangelization of API's in a closed source model basically defaults to trust, OSS API evangelization lets the developer make up his own mind” (Raymond, 1998a). In practice, this perfect method may also be considered a weakness. Valloppillil says, “Whereas the ‘enthusiast developer’ is comforted by OSS evangelization, novice/intermediate developers -- the bulk of the development community -- prefer the trust model + organizational credibility (e.g. ‘Microsoft says API X looks this way’)” (Raymond, 1998a) Whether this is true remains to be proven, but the fact that it might be true, in some cases, could be characterized as a weakness of open source software. In short, what open source software must deal with is the *perception* of problems with its organizational structure and processes. Often large companies and libraries are more comfortable dealing with large software companies. Perhaps there is an assumption of stability and accountability that reassures those implementing the new software or perhaps there is just a feeling of shared values.

Many would argue that the ability to “kick the tires” of a program is enough, that one cannot know a program is good without the ability to view its source code, but there may be some who, for whatever reason, prefer to trust a large corporate body to be working for their best interests. This is something that large GNU/Linux distributors are trying to address by providing a corporate entity from which a contract can be purchased and into with whom a business arrangement can be entered. There are, at present, no library specific, open source businesses, like this, that could reassure a library with these fears and uncertainties. There might be in the future, but Open Source, as a concept, is too new to the library community for the author to be able to assert with any confidence that there one day will be. To address this fear that many large libraries have, this paper investigates open source projects that are not library specific, but that have long standing track records as reliable software.

There are many open software projects that libraries use, often without even realizing they are using, open source software. These programs are often Internet based, have a long history of successful use, and are generally recognized by the computing community as stable, reliable, and well supported by their developer communities. Examples of such open source projects include BIND, Perl, Apache, and Linux. If a library is connected to the Internet, more than likely it is using at least two or three of the above in one way or another. The last, Linux, is probably installed in the library even if the administration or system administrators are unaware of it and some libraries, like the University of North Carolina at Chapel Hill’s Law Library, use Linux as their full-time web server.

BIND is an open source program that any library on the Internet probably uses. BIND, the Berkeley Internet Name Domain package, is software that allows one computer to find another without having to know its unique Internet Protocol (IP) numeric address. BIND, in other words, translates 10.0.1.100 into www.mydomain.com, and vice versa, so that humans can type www.mydomain.com rather than having to remember the IP address, 10.0.1.100. The Internet depends on BIND to function correctly. BIND provides DNS (Domain Name Service) for the entire

Internet and, by extension, for any library connected to it. BIND has been in use since 1984.

Perl is another open source project that is widely used. A scripting language, Perl is responsible for much of the active content on the World Wide Web. It has been called the “duct tape of the Internet.” Perl has been in use since 1987 and is most commonly used for CGI scripting, accessing databases, text processing, XML processing, system administration, web transactions, and many other activities. Even after all these years, Perl is one of the most popular programming languages on the web (Perl Mongers, 2000).

Lastly, Apache, an open source web server, is more than likely the web server most libraries are using to serve their library’s web site, assuming libraries select a web server for the same reasons other site owners do. Since January, 1998, Apache has been the number one web server used on the World Wide Web, as noted in a survey by Netcraft; the survey’s authors observe that Apache is used on over fifty percent of the World Wide Web’s web sites (Apache Software Foundation, 1998). Since that time, the popularity of the Apache web server continues to grow. A new Netcraft survey, published in March, 2000, shows that the Apache web server now has a sixty percent share of the market and, with newly formed web sites, Apache is chosen two out of three times as the site’s new web server (Speedie, 2000).

Open source software is being used by more and more companies and organizations with a presence on the World Web Wide. The Open Source Initiative, in an attempt to promote the reputation of open source software, has posted a list of companies that make at least a million dollars a year that use, or have developed their corporate strategy around, open source software. Some of the names on the list include IBM, Cygnus Solutions, SGI, Apple Computer, Netscape Communications, Inc., and others (Open Source Initiative, 1999a). In addition to the wide variety of reliable open source products available for general use, more and more library specific open source products are being developed by librarians in attempt to solve library specific software problems.

Existing Closed-source Library Software

Software plays an important part in many libraries' daily activities. Often, patrons search for books in an online public access catalog, browse Internet sites, request books from other libraries across the country (many times after searching that library's online catalog via the Internet), and search external databases to which their library provides access. Library staff catalog books, create library and institution specific web pages, satisfy interlibrary loan requests, create pathfinders, and collaborate with colleagues across the country to better serve their patrons. Much of this is done with commercial or free, but closed-source, software.

Most libraries that use computers probably use Microsoft Windows as their client-side operating system for library staff. Much of the word processing that is done in these libraries is probably done on Microsoft Word and library staff probably use Access or Excel to create small databases and spreadsheets for managing the library's expenses or monthly statistics. There are, of course, open source alternatives to all these but, as of yet, none have really made much inroads into the mainstream library. It is likely that the library's web server is running an open source operating system or that the system administrator uses GNOME, or another open source desktop, but the average cataloger, reference librarian, or administrator most likely uses closed-source software. Why?

One reason is that client side software is usually more complex to write. There are client side open source projects under development, but the majority of open source software was written to run on a server. A recent survey of Linux developers revealed that "Only 16 percent write office app[lication]s and only 15 write graphics or audio/video app[lication]s" (Speedie, 2000a). With such a small number of client side applications under development it is little wonder that most librarians are using closed-source software. The author of the survey reports that this is likely to change, however, in part due to the growth of open source desktop projects like KDE and GNOME (Speedie, 2000a).

Another reason why many libraries use closed-source software is that it is free. The Online Computer Library Center (OCLC), a non-profit, membership driven, international library organization, gives away cataloging, interlibrary loan, and reference software to libraries that use, and pay for, its services (OCLC, 2000). OCLC started in 1967 as the Ohio College Library Center and as it grew it developed into an international consortium that enables the sharing of bibliographic records on a world wide scale. The software that OCLC gives away includes the CatME (Cataloging MicroEnhancer), ILL ME (the InterLibrary Loan MicroEnhancer), and Passport (a full featured 16 bit cataloging utility). Libraries who belong to OCLC are able to use these useful library specific tools free of charge. Unfortunately for libraries that cannot afford OCLC's membership fees, the software is restricted to OCLC members. Also, these programs are closed-source; individual libraries must rely on OCLC for updates. Customization is possible, to a limited degree through the OCLC Macro Language (OML), but the basic functionality of these programs is controlled by OCLC.

In addition to the smaller applications used by library staff, many libraries use and maintain an integrated library system that provides a public access catalog for patrons to search and back-end modules for library staff to edit and create bibliographic records, manage the acquisitions of new materials, and monitor the circulation of the library’s collection. There are many third party integrated library system vendors. Some of the most well known include Geac, Data Research Associates (DRA), Ex-Libris, CARL, Gaylord, PALS, SIRSI, and Innovative Interfaces, Inc. At this time, none of these vendors are providing open source software.

Once a library purchases an online public access catalog from one of these vendors, it has entered into a long term contractual agreement with that company, not necessarily because this is what vendors require but because of the amount of effort required for a library to switch vendors. Each vendor relies on a proprietary, closed-source method of storing the library’s records. If a library were to change vendors, as occasionally happens, the entire database would need to be converted. This is a time consuming and rarely trouble-free process. As a result, libraries often select a single integrated library system vendor and then rely, completely, on that vendor to anticipate and respond to the library’s particular needs. Since one vendor will often sell the same turnkey system to a variety of libraries, response time for changes to a library’s system can be slow. For these, and other reasons, many librarians are considering open source alternatives to traditional library software.

Existing Open Source, Library Specific Software

Perhaps the most well known advocate for open source software in the library community, Daniel Chudnov first brought open source software to the attention of the mainstream library community by writing an article titled “Open Source Software: The Future of Library Systems?” This article, published in the August 1999 issue of *Library Journal*, suggests that “Libraries are in general slightly behind the curve with software; similarly, we are slightly behind the curve with open source software. As the free software vision and culture continue to mature, librarians would be remiss not to serve a major role shaping that culture” (Chudnov, 1999). The article also listed many library related open source projects and gave a list of URLs so that librarians interested in learning more could research the open source phenomenon for themselves. Chudnov, a medical librarian at Yale Medical Library, also maintains the OSS4LIB (Open Source Systems for Libraries) web site and email discussion list along with Gillian Goldsmith Mayman.

The OSS4LIB web site (http://www.oss4lib.org) serves as a clearinghouse for information on open source development within the library community. As of April 8, 2000, there were thirty-two separate open source, library specific projects listed on the OSS4LIB page. These ranged in scope and purpose from attempts to create a next generation integrated library system to online reserve modules to bibliographic tools, current awareness programs, and customized knowledge environments for libraries (Chudnov, 2000a). In addition, the OSS4LIB web site summarizes the recent news from other open source and library science web sites; these include Library and Information Science News (LISNews.COM), Slashdot (Slashdot.ORG), and Freshmeat (Freshmeat.NET), among others.

The OSS4LIB web site also houses a fledgling collection of articles relating to open source software and the library community. Readers of the OSS4LIB web page and subscribers to the email discussion list are encouraged by Chudnov to submit their own. In addition to running the OSS4LIB web site, Daniel Chudnov has created several open source packages for use by libraries: the EDD (Electronic Document Delivery) project, the Librarians Guide to Free Software project (based on DocBook), and JAKE (a Jointly Administered Knowledge Environment for libraries). Of these, JAKE is the most actively developed at this time.

JAKE supports the management and linking of online resources and their locally created descriptions. Chudnov says, “Jake consists of a database containing information about e-resources (including online journals, databases, search interfaces, and textbooks) and how they relate to each other” (Chudnov, 2000). Currently there are 193 databases linked and managed by JAKE. Chudnov, who contacts online vendors about inclusion in JAKE has, for the most part, received a positive response. This is noteworthy because the publishers’ content lists, as a result of being in the JAKE database, would be released under the GNU GPL license. This means anyone could use that data in any manner. Chudnov (2000) says, “The providers we have contacted have mostly been very interested in the project and agreeable to its terms; some have even expressed interest in writing their content lists to a specific format to facilitate JAKE inclusion, and some expressed interest in the potential of using JAKE in their own services.”

One of the benefits to the JAKE project is that integrating various related and unrelated online databases allows for a certain level of authority control of the contents to be implemented. A result of this is that patron searches can succeed regardless of the form of the search term. In the past, this has been a problem with crossdatabase searching. JAKE also incorporates a database comparison feature. This allows patrons to determine which databases index a particular article and which might have that article in full-text. To measure the growing popularity of JAKE, Chudnov keeps a graph of patron use, measured by hits; this graph can be accessed from the JAKE web page (Chudnov, 2000). Also on the web page are links to other libraries that are using the JAKE program.

There are many other types of free software projects available on the OSS4LIB web site. There are programs written to solve specific problems, conversion programs, information management programs, and many others. Often, librarians will create their own software to solve their own particular needs. Since libraries share many of the same problems, releasing home grown software to the library community allows for a collaboration that fosters the development of a solution, or solutions, to a shared problem. One example of this is the FreeReserves project. There is another open source, electronic reserve programs available on OSS4LIB but, since FreeReserves was the first (Nakerud, Dawson-Schmidt, and Van Cleave, 1999), this paper will focus on it.

FreeReserves started as a “proof of concept” for a paper written to explore the strengths and weaknesses of a hypothetical home grown electronic reserves system versus a turnkey one. The paper was written from the perspectives of two libraries: one that implemented a home grown solution and one that chose a turnkey system. The paper does a good job at describing the strengths and weaknesses of each approach. It is interesting to note, however, that the library that chose to develop their own electronic reserves system has modified the system to correct many of the weaknesses noted in the paper. The ability to do this is, obviously, one of the biggest benefits of open source software. Shane Nakerud (1999), author of the paper, says:

> Finally, the biggest difference between homegrown and turnkey systems lies in the customizability of the end product. Libraries developing homegrown systems can conduct usability testing to determine the most appropriate web interface, or use existing graphics and web site appearance to create a satisfactory e-reserves site. Anything from graphics, to screen layout, to navigation can all be customized to meet library web page standards or the desires of your library's webmaster or web committee. By creating a homegrown site, SIUC was not only able to customize the patron interface, but also the backend database in order to create a system which totally catered to the needs of their reserve unit.

This was not the only difference, though. Price was eleven thousand for the turnkey system while only around three thousand for the home grown system's hardware, plus an indeterminate amount for the labor involved (Nakerud, 1999). Overall, including the total cost of labor and management, the home grown, open source, solution was more cost effective.

This is not to say that open source software does not have its weaknesses. The University of Minnesota, the library that decided to go with the turnkey system, says that the lack of technical skills among library staff members was the main reason for the decision (Nakerud, 1999). The fear that libraries do not have the technical staff to maintain or create an home grown system can be discouraging to libraries considering an open source product. For this reason, though, the University of Minnesota is now dependent on a proprietary system's software vendor for any changes that need to be made. Nakerud (1999) says:

> The most obvious limitation concerning the ERes product is the ability to customize the interface or the overall system. The ERes license agreement prohibits any modification of the code without the prior approval of Docutek … Docutek does promise that it will develop an ERes system that meets the wants and needs of

> the school. This may require extra software engineer hours, and possibly an extra fee attached to the overall price.

As of April 9, 2000, both schools are still using the electronic reserves system they selected in March of 1999.

One of the most ambitious new library-related open source projects on the Internet today is the Open Source Digital Library System (OSDLS Community, 1999). The OSDLS, an attempt by a relatively small group of librarians to build a next-generation integrated library system, is, like many other Internet-based open source projects, maintained through an email discussion list and online web site. Founded in March 1999 by Jeremy Frumkin, a meta-data librarian at the University of Arizona, the OSDLS project emphasizes its community-based orientation. The project's white paper (OSDLS Community, 1999a) states: "[The OSDLS] will only succeed if a sufficient number of library programmers, librarians, and library staff are involved in the development of the system." By emphasizing the intrinsically communal nature of open source software, the OSDLS project attempts to ensure that "development doesn't define the functionality and use of the product" (OSDLS Community, 1999a).

Instead of letting development needs shape the evolution of the product, the OSDLS project emphasizes community and end-user participation in an attempt to create and promote a process that is responsive to the needs of its intended user base. By providing access to the OSDLS through the group's web site, FTP server, and email discussion list, the project's development community gives potential users a chance to interact with the OSDLS *before* the finished product is ever shipped. This attempt to engage software users, who may or may not have any development experience, or knowledge of the development process, has proven essential to the successful development of many other open source projects (Raymond, 1999d). OSDLS project members hope that, by modeling their processes after the successes of other open source projects, they will be able to capitalize on the strengths of open source software development.

While the OSDLS community is still relatively small, there seems to be a growing interest in the project and in the development of other open source software alternatives for the library community. For instance, after recent announcements about the existence of the OSDLS project to other library and information science-related mailing lists (ASIS-L, Web4Lib, and DIGLIB), the number of subscribers to the OSDLS mailing list, as reported by Jeremy Frumkin (1999d), experienced a noticeable surge. Whether the growing interest in projects like the OSDLS will, in fact, help produce a marketable product that can be used and further developed by the library community, however, remains to be seen. Given the project’s brief history, its newly formed partnerships with other open source library groups, and its articulated vision of the future of library systems, I believe the OSDLS' prospects are good. This is not to say that the path to a marketable product will be easy. Library systems are complex programs, often containing over a million lines of code (Summerhill, 1999). Whether the OSDLS project will be able to grow from its humble beginnings, as an offshoot of the Linux-in-Libraries email discussion list, to become “a robust alternative to what many perceive as inferior, over-priced products (OSDLS Community, 1999a),” produced by closed-source proprietary library system vendors, still remains to be seen. This paper argues that such an evolution is, at the very least, possible based on the dynamic nature of open source software development.

One of the most obvious benefits of open source development is that one project tends to spawn other complementary projects; the result of this is that new ideas, code and project participants are often shared among several open source projects. The genesis of the OSDLS is a good example of how these types of cooperative efforts happen. The idea for an “open source digital library system” originated in a brainstorming session on the Linux-in-Libraries (LIL) email discussion list. This list was formed to discuss libraries’ use of another open source project, Linux. The discussion on the current state of library systems started on the Linux-in-Libraries listserv when one list member asked about currently available library software for a Linux server. After the idea of an open source digital library system was entertained by several members of the discussion list, Jeremy Frumkin (1999b), a meta-data librarian at the University of Arizona, took the lead in an attempt to map out what such a project would need to do to organize itself. The first step was to create a white paper that would detail the project’s goals; once this was done, he suggested that grant money could be obtained to further develop the project. Frumkin also recommended that the OSDLS group model its organization and development after two other well-known open source software projects, Linux and Mozilla.

In *The Cathedral and the Bazaar*, Eric Raymond suggests that open source projects must, by necessity, always be initiated by a small group of people, if not by individuals. This is true for the OSDLS project just like it is for the many other library open source projects that I have mentioned. At this early stage in the Open Source Digital Library System’s development, Art Rhyno, a systems librarian at the Leddy Library of the University of Windsor, has done most of the coding and database design. As a result, his particular contributions are, as they should be, officially recognized on the OSDLS web site (OSDLS Community, 1999). This is not to say that others haven’t made suggestions or posted code with alternate ways of accomplishing a particular task, or that Art Rhyno hasn’t encouraged others to contribute, but the bulk of the work that is maintained on the OSDLS’ FTP site was created by Art Rhyno. This is not necessarily a bad thing. Raymond (1999e) says, “It's fairly clear that one cannot code from the ground up in bazaar style. One can test, debug and improve in bazaar style, but it would be very hard to originate a project in bazaar mode.” The strength of open source development lies in its ability to refine existing code. All software projects, Raymond believes, originate through the inspiration and initial hard work of an individual. This is true of all projects regardless of whether they are developed using an open source or proprietary style of development (Raymond, 1999b).

How then does one make the leap from an individual’s initial effort to a full-scale open source project? To attract participants to a new open source project, Raymond suggests a charismatic person is needed to organize the initial stages of development. He says, “In order to build a development community, you need to attract people, interest them in what you're doing, and keep them happy about the amount of work they're doing. Technical sizzle will go a long way towards accomplishing this, but it's far from the whole story. The personality you project matters, too” (Raymond, 1999e).

As the founder of the OSDLS project, Frumkin has managed to generate interest in the OSDLS through a variety of methods. He has coordinated his efforts with other prominent open source proponents, of which Dan Chudnov of the Open Source Software for Libraries (OSS4LIB) web page is probably the most well known to the library community. Frumkin is also in the process of publishing an article on the Open Source Digital Library System for a pan-European e-journal funded by the European Commision’s Telemetrics for Libraries Programme (Frumkin, 1999). As already mentioned, he has promoted the OSDLS through other email discussion lists (ASIS-L, Web4Lib, and DIGLIB) and has created and manages a machine dedicated to running the OSDLS web server, FTP server and, eventually, an operating version of the OSDLS (Frumkin, 1999c). The interest Frumkin has created in the OSDLS not only extends to the project’s participants, but also to potential distributors of the OSDLS package. Despite the fact that the project’s code is still in its infancy, the OSDLS community has already been contacted by OpenClassroom, an independent company interested in distributing the OSDLS product to the educational sector (Lacal, 1999). For the relative newness of the OSDLS project, it has managed to garner a good deal of publicity. The question is, will the OSDLS group be able to produce an integrated library system that libraries want to use?

One thing that the OSDLS project has in its favor is that the programmers do not intend to write all the system’s code from scratch. Building a complete integrated library system without using any existing code may seem like an impossible task when one takes into consideration that many current library systems contain well over a million lines of code. There is no reason, however, for an open source project to write that many lines of new code; there are existing library and non-library open source products that can be incorporated into the Open Source Digital Library System with relative ease. One such project is the MySQL database. Ideally, the OSDLS project will not be dependent on a particular database backend, but could just as easily be used with PostreSQL as Oracle 8i. For the time being, however, some of the code used to load MARC records into the system is MySQL-dependant.

Building a free software project around existing open source tools is a popular approach for new projects. Another library project that was just published under the GNU GPL license is the MyLibrary@NCSU project. Unlike the OSDLS, MyLibrary@NCSU has been around for several years as a developmental project at North Carolina State University. The MyLibrary project, created by Eric Morgan, a NCSU Digital Initiatives librarian, is a customizable knowledge environment for the academic library. It was started in 1997 in response to focus group interviews conducted by the NCSU Department of Digital Libraries Initiative (Morgan, 1998). Around that time, Internet sites like Yahoo, Excite, and DejaNews were creating portals to the World Wide Web. MyLibrary@NCSU does this, but it also adds a human component to the environment by allowing MyLibrary users to interact with a real live librarian who can assist them with their information needs.

Eric Morgan, who has created many freeware tools for librarians and computer users in general, believes this type of interaction is essential for library-based open source projects. He says:

> Human interactions are a necessary part of the mixture in both librarianship and open source development. Open source development requires people skills by source code maintainers. It requires an understanding of the problem the computer application is trying to solve, and the maintainer must assimilate patches with the application. Similarly, librarians understand that information seeking behavior is a human process. While databases and many ‘digital libraries’ house information, these collections are really ‘data stores’ and are only manifested as information after the assignment of value are given to the data and inter-relations between datum are created.[2]

[2] Morgan, Eric (eric_morgan@ncsu.edu). (2000, February 9). Subject: Gift cultures, librarianship, and open source software development. Email from author to crivers@clatsop.cc.or.us, arhyno@server.uwindsor.ca, and kevin_clarke@unc.edu.

By adding an opportunity to interact with a trained information services librarian, MyLibrary draws from the strength of the traditional library community emphasis on service and links it with the advantages of open source software.

The response to MyLibrary@NCSU has been impressive. In the first seven days after its release was announced on a few library related email discussion lists, the MyLibrary project was downloaded eighty-one times. Sixty-nine of those downloaders registered their copy of the program and 41 people subscribed to the email discussion list. In addition to those that downloaded the program, one hundred thirty-seven people registered to test out the online version of the program; the online version was visited one thousand nine hundred twenty-six times (Morgan, 2000). That the MyLibrary idea is a good idea also seems to be supported by the fact that many other libraries around the country have also implemented or are researching similar projects; MyLibrary@Cornell (Cornell University, 2000), MyLibrary@CalPoly (California Polytechnic State University, 2000) and VCU’s ‘My Library’ (Virginia Commonwealth University, 2000) are just a few examples.

What distinguishes the North Carolina State Project from the others is that the librarians at NCSU have released the project as open source software. The MyLibrary project is too new, as open source software, for us to predict whether it will be adopted by many other libraries (e.g. whether the open source model will work and other libraries will contribute to its development), but it looks promising. MyLibrary’s email discussion list has had many librarians from a variety of library types posting suggestions or possible improvements to the system. Many ask whether the list thinks that MyLibrary can be applied to their particular situation.

Eric Morgan, the NCSU Digital Initiatives librarian responsible for the management and development of MyLibrary@NCSU has posted a development plan on the MyLibrary web site that, he hopes, will help drive the development and adoption of the MyLibrary project. The plan divides the growth of MyLibrary into three main sections: phase one, the initial adoption by power users, phase two, the stage for beta-testers, and phase three, MyLibrary's general release (Morgan, 1999b). MyLibrary is currently in its initial phase. Since its release, MyLibrary@NCSU has generated many suggestions for improvements, many of which are a result of trying out the system in a variety of environments. This diversity of testing environments would not have been possible if the project was not open sourced.

Also contained in the page that maps out the development of MyLibrary@NCSU is a proposal for a licensing plan under which the MyLibrary project is to be released. The licensing plan calls for MyLibrary to be released under one of two licenses (at the option of the licensee). The first would give libraries permission to use MyLibrary without the promise of any support from NCSU. The second, which is a co-development license, gives permission for other libraries to obtain the source code and contribute to the project. Under this second license, NCSU would supply support for the product because the licensing library would be collaborating with NCSU in the development of the project (Morgan, 1999b).

On January 27, 2000, Eric Morgan posted an email to the OSS4LIB mailing list to discuss the possibility of releasing MyLibrary@NCSU under the GNU GPL license. He says:

> I have just finished reading Eric S. Raymond, *The Cathedral And The Bazaar,* … and I believe the principles of the open source movement are similar to the principles of librarianship. What do you think? In the NCSU Libraries we

> are seriously considering sharing the source code to MyLibrary@NCState under the GPL. Hopefully we will improve upon the MyLibrary by using the open source development model (Morgan, 2000a).

Morgan continues by asking what the list thinks of open source software and its potential for use in the library setting. He then summarizes three characteristics that he believes are similar to the open source and library communities. Both are gift cultures; both depend on human interactions for their success; and both are experiencing an increased demand for capable practitioners (in part because of the abundance of resources within the community, a characteristic of a gift culture).

MyLibrary@NCSU is unique because, unlike the other open source library specific projects mentioned in this paper, it was not created as an open source project, but became one as the library began to investigate different licensing options. This suggests that open source software model offers certain tangible benefits in addition to the ideological ones that often motivate programmers to create open source projects. This suggestion is supported by a newly released study, *Strategies for the Knowledge Economy: From Rhetoric to Reality*. Though the study refers to businesses, it acknowledges that the times are being shaped by the Internet/open source community’s culture.

The study prepared by the executive search firm Korn/Ferry International, in cooperation with the University of Southern California's Center for Effective Organizations at the Marshall School of Business, says that for businesses to stay viable in the current information market they must adopt some form of information sharing. The study explains that the old “cloak and dagger” style of information management is no longer successful and that an open, fast paced “Internet-style” of information sharing is the only way for businesses to stay competitive (Sharett, 2000). If this is true, one wonders whether library software vendors will adopt this tactic in an attempt to remain relevant to the library community.

One major library software company already has. It is not producing open source software, but it has designed its integrated library system to run on GNU/Linux, an extremely popular open source operating system. About their decision to be the first automation vendor in North America to support the GNU/Linux operating system, they say:

> In evaluating the available technology for these types of libraries, we carefully considered various operating systems. Our studies, using the current versions of these operating systems, showed that Linux could best address performance, scalability, and affordability at this point in time. Furthermore, the level of functionality was an important consideration, and it was found that no loss of functionality occurred as a result of using Linux (Ex Libris, 2000).

It is unlikely that Ex-Libris will be the last library software vendor to support GNU/Linux, given its rapid growth in the server market, but whether existing library software companies will produce their own open source software remains to be seen.

This paper has given examples of software companies, like Cygnus Solutions, that have succeeded by developing open source software but for current library software companies to do this would require a change in the structure and purpose of these companies. It would require them to view themselves as service providers, like the libraries they sell to, rather than as manufacturers of library software. It may be more likely, as the Keystone Principles have suggested, that libraries will build their own open source solutions to the problems that plague the proprietary software of today. These solutions could then be improved and promoted by the library community itself, removing the middlemen and with them a large portion of the cost of maintaining an integrated library system.

The Future of Library Specific, Open Source Software

At this time, there is a great deal of possibility for library specific, open source software. There is community support and support at the national level. There are also individual projects that are doing well and attracting the attention of many in the library community. There is no evidence, though, that library specific, open source software will succeed in the long run. Library specific, open source software will, without a doubt, always exist, because librarians will always be creating solutions to their software problems (what Raymond calls “scratching their own itches”), but whether open source software will replace the proprietary, closed-source software that libraries currently use remains to be seen. Daniel Chudnov, author and maintainer of the Open Source Systems for Libraries web page, says, “For now, open source software has not made major inroads in the library market, but some projects suggest great possibility” (Chudnov, 1999). The question becomes do librarians, as a community, support the effort to create open source software because it spawns from a tradition similar to our own and offers us economic advantages that our current proprietary software does not?

This paper has suggested that we should. There is support at the national level and there are individual projects underway. Open source software would allow for a peer-review process that would ensure that the library systems of the future are not co-opted by commercial interests that skew the results of a patron’s search based on advertising revenue. Open source software would allow for libraries to customize their system software to match their patrons' information needs unlike proprietary software that specifically forbids this. As Raymond and others have documented, open source software also offers more security and a rapid evolution cycle because of the collaboration that occurs between a variety of programmers from different institutions and locations around the world. Lastly, as Shane Nakerud observed, the total cost of owning open source software is lower than that of proprietary software, though "free software" is not free from cost.

Libraries that accept open source software will experience a shift in their budgeting. Where once they bought integrated library systems from offsite vendors, now they will spend money on librarians trained to integrate the library's needs into freely available shared software. The thought of hiring more librarians has discouraged some from accepting open source solutions to their problems; instead, they preferred to pay larger sums to software companies in exchange for turnkey systems that must be maintained through a contractual vendor relationship, year after year. Some library administrators are concerned they will not be able to get support for a system that does not cost the library any money. These and other concerns have been answered by Eric Raymond (1998a) in response to an internal Microsoft document that was leaked from the company. In addition, the existence of the Internet, with its reliance on BIND, Perl, Sendmail, and Apache is proof that open source software works and has a well-supported user base.

Though the success of open source software is not a new development, its potential has recently reached the awareness of the mainstream media and public. Given the recent meteoric rise in use and popularity of open source software, it seems just a matter of time before library specific, open source projects attract more attention in the library community. At this juncture, librarians and library administrators have the choice whether to officially support the development of library specific open source software, like the library administrators who met in Keystone Colorado have suggested, or to let these projects develop on their own. If the decision is made to reap the benefits of this collective movement, libraries will; if the choice is made not to encourage their growth, it is likely that individual projects will remain just individual librarians’ solutions to their libraries’ problems.

This paper suggests that libraries encourage the growth of library specific, open source software in the library community and then, once it reaches a mature state, study it. Open it up to review and compare the results of library specific, open source projects with the results of the proprietary software written for libraries. Which is more responsive to the needs of librarians? Which has a lower total cost? Which evolves based on a particular library’s patron feedback? Which is more customizable? These are all, in the opinion of the author, directions for future research that would help document the success of community based software development.

Bibliography

American Library Association (1995). Code of ethics of the American Library Association. *Office for Intellectual Freedom Documents* [Online]. Available: http://www.ala.org/alaorg/oif/ethics.html [2000, March 27].

American Library Association (1997). Questions and answers: Access to electronic information, services, and networks: An interpretation of the Library Bill of Rights. *Interpretations of the Library Bill of Rights* [Online]. Available: http://www.ala.org/alaorg/oif/oif_q&a.html [2000, March 27].

American Library Association (1999). Libraries, an American value. *Office for Intellectual Freedom Documents* [Online]. Available: http://www.ala.org/alaorg/oif/lib_val.html [2000, March 27].

American Library Association (2000). Equity of the information superhighway. *Library Advocacy Now! Action Pack* [Online]. Available: http://www.ala.org/advocacy/action/act1.html [2000, March 27].

Apache Software Foundation (1998). Apache webserver serves over half the Internet. *Apache Home Page* [Online]. Available: http://www.apache.org/press/05Jan98.txt [2000, April 16].

Association for Research Libraries and Online Computer Library Center (1999). The Keystone Principles. *OCLC Institute News* [Online]. Available: http://www.oclc.org/institute/keystoneprinciples.htm [2000, April 15].

Belkin, N., Oddy, R., & Brooks, H. (1982). ASK for information retrieval. Part 1: Background and theory. *Journal of Documentation,* 38(2).

Buckland, M. (1983). *Library Services in Theory and Context*. New York: Pergamon.

California Polytechnic State University (2000). MyLibrary research page. *Robert Kennedy Library* [Online]. Available: http://www.lib.calpoly.edu/mylib/cgi-bin/index.cgi [2000, April 15].

Clarke, K.S. (1999). PyAdd.java hack. *Open Source Digital Library System Listserv* [Online]. Available: http://listserv.arizona.edu/cgi-bin/wa?A2=ind9909&L=osdls&P=R2&D=0 [2000, April 16].

Cornell University (2000). CU's MyLibrary is personalized web interface to networked resources. *Cornell Chronicle* [Online]. Available: http://www.news.cornell.edu//Chronicles/2.10.00/MyLibrary.html [2000, April 15].

Chudnov, D. (1999). Open source software: The future of library systems? *Library Journal*, 124(13).

Chudnov, D. (2000). *Jointly Administered Knowledge Environment (JAKE)* [Online]. Available: http://jake.med.yale.edu/docs/about.html [2000, April 9].

Chudnov, D. (2000a). Projects. *Open Source Systems for Libraries (OSS4LIB)* [Online]. Available: http://www.oss4lib.org/projects/welcome.php [2000, April 8].

Corcoran, C. (1997). Are we ready for the library of the future? *Salon* [Online]. Available: http://www.salon.com/21st/feature/1997/12/02feature.html [2000, April 15].

Danton, J. (1975). Plea for a philosophy of librarianship. *American Library Philosophy: An Anthology*, ed. B. McCrimmon. Hamden, CN: Shoe String Press, Inc.

DiBona, C., Ockman, S. & Stone, M. (1999). Contributors. *Open sources: voices from the Open Source revolution* [Online]. Available: http://www.oreilly.com/catalog/opensources/book/authors.html [2000, March 22].

Dougherty, D. & Sims, D. (2000). Will money spoil Open Source? *Linux DevCenter* [Online]. Available: http://oreilly.linux.com/pub/a/linux/2000/01/31/interview/index.html [2000, March 21].

Encyclopedia of Library and Information Science: Vol. 25. (1968). New York: Marcel Dekker, Inc.

Ex Libris (2000). Aleph 500 to run on Linux! *Press Releases* [Online]. Available: http://www.aleph.co.il/news1.asp?categoryId=61&admin= [2000, April 16].

Frumkin, J. (1999). ALA open source gathering. *Open Source Digital Library System* [Online]. Available: http://osdls.library.arizona.edu/ALA/ [1999, October 20].

Frumkin, J. (1999b). LIL--Open Source Digital Library System. *Linux-in-Libraries Listserv* [Online]. Available: http://edvmix3.ub.tu-berlin.de/lists/linux-in-libraries/199903/19990306.html#3 [1999, October 20].

Frumkin, J. (1999c). OSDLS: Open Source Digital Library System. *Open Source Digital Library System* [Online]. Available: http://osdls.library.arizona.edu/ [1999, October 20].

Frumkin, J. (1999d). OSDLS update. *Open Source Digital Library System Listserv* [Online]. Available: http://listserv.arizona.edu/cgi-bin/wa?A2=ind9905&L=osdls&P=R154 [1999, October 19].

Hildreth, C. (1982). The concept and mechanics of browsing in an online library catalog. *Proceedings of the 3rd National Online Meeting*. Medford, NJ: Learned Information, Inc.

Indiana University (1998). *IU School of Information and Library Science* [Online]. Available: http://www.iupui.edu/home/libinf.html [2000, April 14].

Kilpatrick, T. (1997). ALA reports--San Francisco, June 26 - July 3, 1997 Social Responsibilities Round Table. *Southern Exposure: The Newsletter of Morris Library* [Online]. Available: http://www.lib.siu.edu/southex/1997/jul17.html [2000, April 16].

Kochen, M. (1993). Information and society. *Annual Review of Information Science and Technology*, 18. White Plains, NY: Knowledge Industry Publications.

Lacal, J. (1999). Re: Thoughts on making the OSDLS a distribution. *Open Source Digital Library System Listserv* [Online]. Available: http://listserv.arizona.edu/cgi-bin/wa?A2=ind9909&L=osdls&P=R269 [1999, October, 20].

Lehey, G. (1999). The path ahead. *The Daemon News: Bringing BSD Together* [Online]. Available: http://www.daemonnews.org/199912/d-advocate.html [2000, March 25].

Leonard, A. (1998). Let my software go. *Salon* [Online]. Available: http://www.salon.com/21st/feature/1998/04/cov_14feature.html [2000, April 16].

LinuxDevices.Com (1999). Red Hat + Cygnus merger creates Open Source powerhouse. *LinuxDevices.Com: the Embedded Linux Portal* [Online]. Available: http://www.linuxdevices.com/cgi-bin/news_view.cgi?newsid=NS6531911783 [2000, March 21].

Lyman, P. (1997). Digital documents and the future of the academic community. *Scholarly Communication and Technology* [Online]. Available: http://arl.cni.org/scomm/scat/lyman.html [2000, March 26].

Merriam-Webster (2000). Potlatch. *Merriam-Webster Online Dictionary* [Online]. Available: http://search.eb.com/cgi-bin/dictionary?va=potlatch [2000, March 27].

Morgan, E. (1998). About MyLibrary. *About* MyLibrary@NCSU [Online]. Available: http://my.lib.ncsu.edu/about/about-1.1.2b/ [2000, April 9].

Morgan, E. (1999). Marketing future libraries. *Publications* [Online]. Available: http://www.lib.ncsu.edu/staff/morgan/cil/marketing/index.html [2000, March 27].

Morgan, E. (1999a). Marketing through usability. *Publications* [Online]. Available: http://www.lib.ncsu.edu/staff/morgan/cil/usability/index.html [2000, March 27].

Morgan, E. (1999b). MyLibrary co-development and source code distribution plan. *About MyLibrary@NCSU* [Online]. Available: http://hegel.lib.ncsu.edu /development/mylibrary/about/distribution-plan.html [2000, April 15].

Morgan, E. (2000). Just for the record. *MyLib-Dev Mailing List* [Online]. Available: http://hegel.lib.ncsu.edu/development/mylibrary/support /mailing-list/0031.html [2000, April 15].

Morgan, E. (2000a). Open Source and librarianship. *Open Source Systems for Libraries Listserv* [Online]. Available: http://www.oss4lib.org/listserv/msg00122.php [2000, March 27].

Morgan, E. (2000b). Personalized library interfaces. *MyLib-Dev Mailing List* [Online]. Available: http://hegel.lib.ncsu.edu/development/mylibrary /support/mailing-list/0087.html [2000, April 14].

Mukherjee, A. (1966). *Librarianship: Its Philosophy and History*. Bombay: Asia Publishing House.

McHugh, J. (1998). For the love of hacking. *Forbes Magazine* [Online]. Available: http://www.forbes.com/forbesglobal/98/0810/0109044a.htm [2000, March 20].

Nakerud, S. (1999). E-Reserves: Home grown vs. turnkey. *The Paper—FreeReserves* [Online]. Available: http://www.lib.umn.edu/san/freereserves/paper.html [2000, April 9].

Nakerud, S., Dawson-Schmidt, P. & Van Cleave, K. (1999). *FreeReserves* [Online]. Available: http://www.lib.umn.edu/san/freereserves/ [2000, April 9].

NetBSD Community (1999). BSD community welcomes Apple's new Open Source operating system.” *News Release* [Online]. Available: http://www.netbsd.org/gallery/press/19990607a.html [2000, March 26].

OCLC (2000). *OCLC Access Suite* [Online]. Available: http://www.oclc.org/oclc/menu/suite/index.htm [2000, April 24]

Open Source Initiative (1999). Approved licenses. *Open Source Software Gets Honest* [Online]. Available: http://www.opensource.org/licenses/ [2000, March 26].

Open Source Initiative. (1999a). Open Source Products. *Open Source Initiative Home Page* [Online]. Available: http://www.opensource.org/products [2000, April 16].

OSDLS Community (1999). OSDLS: About the OSDLS. *Open Source Digital Library System* [Online]. Available: http://osdls.library.arizona.edu/about.html [1999, October 24].

OSDLS Community (1999a). The OSDLS white paper. *Open Source Digital Library System* [Online]. Available: http://osdls.library.arizona.edu/OSDLSW.html [1999, October 19].

Perl Mongers (2000). Fast perl facts. *Perl Mongers Press Room* [Online]. Available: http://www.perl.org/press/fast_facts.html [2000, April 16].

Rauch, E. (1998). Mentions of ‘Open Source’ counted by Lexis/Nexis Search. *History of the Open Source Initiative* [Online]. Available: http://www.opensource.org/graphics/mentions.png [2000, March 15].

Raymond, E. (1998). The hacker milieu as a gift culture. *Homesteading the Noosphere* [Online]. Available: http://www.tuxedo.org/~esr/writings/homesteading/homesteading-6.html [2000, April 22]

Raymond, E. (1998a). Open source software: A (new?) developmental methodology. *The Halloween Documents* [Online]. Available: http://www.opensource.org/halloween/halloween1.html [2000, April 16].

Raymond, E. (1998b). Varieties of hacker ideology. *Homesteading the Noosphere* [Online]. Available: http://www.tuxedo.org/~esr/writings/homesteading/homesteading-2.html [2000, March 26].

Raymond, E. (1999). A brief history of hackerdom. *Open Sources: Voices from the Open Source Revolution* [Online]. Available: http://www.oreilly.com/catalog/opensources/book/raymond.html [2000, March 12].

Raymond, E. (1999a). The cathedral and the bazaar. *The Cathedral and the Bazaar* [Online]. Available: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar-1.html [2000, March 23].

Raymond, E. (1999b). The cathedral and the bazaar: Footnotes. *The Cathedral and the Bazaar* [Online]. Available: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar-15.html#[IN] [1999, October 24].

Raymond, E. (1999c). *The History of the Open Source Initiative* [Online]. Available: http://www.opensource.org/history.html [2000, March 26].

Raymond, E. (1999d). The importance of having users. *The Cathedral and the Bazaar* [Online]. Available: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar-3.html [2000, March 23].

Raymond, E. (1999e). Necessary preconditions for the bazaar style. *The Cathedral and the Bazaar* [Online]. Available: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar-9.html [1999, October 20].

Raymond, E. (1999f). Release early, release often. *The Cathedral and the Bazaar* [Online]. Available: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar-4.html [2000, April 8].

Raymond, E. (1999g). The social context of open source software. *The Cathedral and the Bazaar* [Online]. Available: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar-10.html [2000, April 9].

Raymond, E. (1999h). When is a rose not a rose. *The Cathedral and the Bazaar* [Online]. Available: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar-5.html [2000, March 26].

Raymond, E., ed. (2000). The jargon file. *Jargon File Resources* [Online]. Available: http://www.tuxedo.org/~esr/jargon/jargon.html [2000, March 13].

Richardson, E. (1975). The book and the person who knows the book. *American Library Philosophy: An Anthology*, ed. B. McCrimmon. Hamden, CN: Shoe String Press, Inc.

San Francisco Public Library (1998). Internet use policy. *San Francisco Public Library* [Online]. Available: http://nora.sfpl.lib.ca.us/www/internet.html [2000, April 15].

Sharett, K. (2000). Information sharing is key to corporate success. *SmallCapCenter.Com* [Online]. Available: http://www.smallcapcenter.com/story.asp?storytype=sccnews&component=story.asp&storyid=6620 [2000, April 16].

Shera, J. (1970). *Sociological Foundations of Librarianship*. Bombay: Asia Publishing House.

Software Magazine. (1999). The software 500. *Software Magazine* [Online]. Available: http://www.softwaremag.com/archive/1999/0699alpha.html [2000, March 21].

Speedie, A. (2000). Apache grabs two-thirds of new websites. *Wide Open News* [Online]. Available: http://www.wideopen.com/story/670.html [2000, April 16].

Speedie, A. (2000a). What Are Linux Developers Thinking?! *Wide Open News* [Online]. Available: http://www.wideopen.com/story/680.html [2000, April 22]

Stallman, R. (1991). GNU General Public License. *GNU Project--Free Software Foundation* [Online]. Available: http://www.fsf.org/copyleft/gpl.html [2000, March 20].

Stallman, R. (1998). The GNU Project. *The Philosophy of the GNU Project* [Online]. Available: http://www.gnu.org/gnu/the-gnu-project.html [2000, March 12].

Stallman, R. (1998a). Selling free software. *The Philosophy of the GNU Project* [Online]. Available: http://www.fsf.org/philosophy/selling.html [2000, March 12].

Stallman, R. (1999). Free Software Foundation. *Free Software Foundation* [Online]. Available: http://www.fsf.org/fsf/fsf.html [2000, March 12].

Stallman, R. (1999a). What is the copyleft? *Free Software Foundation* [Online]. Available: http://www.fsf.org/copyleft/copyleft.html [2000, March 20].

Stallman, R. (1999b). Why ‘Free Software’ is better than ‘Open Source’. *Free Software Foundation* [Online]. Available: http://www.fsf.org/philosophy/free-software-for-freedom.html [2000, March 26].

Stallman, R. (2000). What is Free Software? *Free Software Foundation* [Online]. Available: http://www.fsf.org/philosophy/free-sw.html [2000, April 22].

Sullivan, B. (1999). BSD a better OS than Linux? *ZDNet News* [Online]. Available: http://www.zdnet.com/filters/printerfriendly/0,6061,2299366-2,00.html [2000, March 26].

Summerhill, C. (1999). Re: LIL--Electronic Card Catalog. *Linux-in-Libraries Listserv* [Online]. Available: http://edvmix3.ub.tu-berlin.de/lists/linux-in-libraries/199903/19990304.html [1999, October 24].

Taylor, R. (1968). Question negotiation and information seeking in libraries. *College & Research Libraries,* 29.

Tiemann, M. (1999). Future of Cygnus Solutions: An entrepreneur’s account. *Open Sources: Voices from the Open Source Revolution* [Online]. Available: http://www.oreilly.com/catalog/opensources/book/tiemans.html [2000, March 20].

Time Magazine (2000). The Open Source revolution newsfile. *Time Digital* [Online]. Available: http://www.time.com/time/digital/reports/opensource/index.html [2000, March 26].

Torvalds, L. (1999). The Linux edge. *Open Sources: Voices from the Open Source Revolution* [Online]. Available: http://www.oreilly.com/catalog/opensources/book/linus.html [2000, March 22].

Vaughan-Nichols, S. (1999). The oldest free OS. *ZDNet Sm@rt Reseller: Opinion* [Online]. Available: http://www.zdnet.com/sr/stories/column/0,4712,398025,00.html [2000, March 26].

Virginia Commonwealth University (2000). About My Library. *VCU Libraries* [Online]. Available: http://www.library.vcu.edu/mylibrary/about.html [2000, April 15].

Appendix A: Open Source Licenses

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the

announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special

The BSD License

The following is a BSD license template. To generate your own license, change the values of OWNER, ORGANIZATION and YEAR from their original values as given here, and substitute your own.

Note: The advertising clause in the license appearing on BSD Unix files was officially rescinded by the Director of the Office of Technology Licensing of the University of California on July 22 1999. He states that clause 3 is ``hereby deleted in its entirety.''

Note the new BSD license is thus equivalent to the MIT License, except for the no-endorsement final clause. <OWNER> = Regents of the University of California <ORGANIZATION> = University of California, Berkeley <YEAR> = 1998

In the original BSD license, the first occurrence of the phrase "COPYRIGHT HOLDERS AND CONTRIBUTORS" in the disclaimer read "REGENTS AND CONTRIBUTORS".

Here is the license template:

The "Artistic License”

Preamble

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

Definitions:

"Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.

"Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder.

"Copyright Holder" is whoever is named in the copyright or copyrights for the package.

"You" is you, if you're thinking about copying or distributing this Package.

"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a) place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said

The End