The Thick of the Fray: Open Source Software in Libraries in the First Decade of this Century

by K.G. Schneider

Open Source Software in Libraries

The library community has been a-buzz of late about open source software (OSS). The air is thick with wildly divergent opinions of its value and utility for libraries even as wikis, blogs, conferences and journal articles about OSS flood the library attention-economy.

The Body Count

To no one's surprise, a raw "body count" of libraries using open source integrated library systems indicates that the vast majority of libraries continue to rely on legacy proprietary systems. However, awareness of OSS as a potentially viable approach for library technology is much more widespread and growing, particularly as libraries use and become comfortable with reliable, high-performing OSS programs such as Firefox.

Libraries are even reengaging with software development projects for the first time in decades, and there are at least a dozen active OSS projects based in or with their genesis in library organizations, including such better-known projects as the integrated library systems Evergreen, Koha, OPALS, OLE and xCatalog; discovery layers such as LibraryFind, Blacklight, VuFind, Fish4Info, Scriblio and SOPAC; and component projects such as Umlaut, the OpenURL resolver and iVia, a search engine/portal.

Major grants through LSTA and Mellon have helped fuel some of these projects, while library systems committed to OSS development have donated sweat equity, hardware and the gift of moral support.

Even OCLC – an organization which for years has zealously protected its data and which in a public relations faux pas in the 1990s sicced its lawyers on New York's Library Hotel for using Dewey decimal numbers on its hotel rooms – has gotten into the game. OCLC's Developer's Network now collaborates in an "open source, code-sharing infrastructure" in which library developers, among other things, "share software code with other network members and the community-at-large in an open source environment" [1].

K.G. Schneider, community librarian with Equinox Software, Inc., can be reached by email at kgs<at>freerangelibrarian.com

What Is Free?

The chattering classes have been busy with OSS. At one extreme, skeptics condemn OSS as a low-grade, poor-man's substitute for licensed software and argue that libraries are better served with off-the-shelf proprietary products. At the other extreme, OSS evangelists intimate it can cure cancer, dissolve cellulite and always out-perform any other software. Between both extremes float misconceptions, assumptions, anxieties and wish fulfillment.

Somewhere amid the fog and friction of myth lie the simple facts of open source software. What makes OSS different from proprietary software is that it is free in every sense of the word: free as in "no cost," free as in "unencumbered" and free as in "not locked up." OSS is free to download, free to use and free to modify (though the process for committing modifications to the core code has a slight air of mystery, as will be discussed below). OSS code is also freely viewable.

Each definition of "free" has its own significance for the value of OSS in libraries, and each definition has also led to some confusion.

For software to be free as in "no cost" means that the software itself has no licensing fees. This definition of "free" has led library technologists such as Eric Lease Morgan to observe that free software is free as in "free kittens" – that is, that these programs still require support and maintenance. It is a matter of debate how freedom from licensing fees directly factors into the total cost of ownership for software.

Marshall Breeding, the director of Innovative Technologies and Research

NEXT PAGE >

for the Jean and Alexander Heard Library at Vanderbilt University and author of the popular website, Library Technology Guides, has frequently questioned whether OSS is overall less expensive than its proprietary counterparts and has called for libraries to look hard at cost factors. He tells me that it is important to question whether OSS offers a lower overall TCO (total cost of ownership) than its proprietary counterparts and to not base decisions on philosophical preferences.

Meanwhile, organizations such as the Georgia Public Library Service (GPLS) have reported significant first- and second-year savings in the implementation and maintenance costs for their automated systems. In the case of GPLS, funding from other organizations has underwritten all or part of new services for their Evergreen system, including acquisitions, internationalization, a low-cost non-SIP (session initiative protocol) self-check alternative and other features.

The "free-as-in-no-cost" characteristics of OSS have led to concerns that libraries would have to provide their own support for OSS. This is a legitimate concern, as few libraries (or for that matter, few organizations in any profession) are in a position to provide most or all of the support and development required to maintain and develop software.

However, "free-as-in-no-cost" does not preclude commercial support models for OSS – and in fact, unlike the proprietary software world, because OSS is open to anyone to use, this vaguely communistic approach to software development has led to a strong free market for software support and development services. In the world beyond libraries, numerous companies have arisen to provide support for various OSS products, such as Red Hat for Linux and Acquia for Drupal.

The Net Under the Tightrope

Within LibraryLand, at least five companies now provide support for OSS. Several of these companies are associated with a specific software program (Equinox for Evergreen, Liblime for Koha and Media Flex for OPALS), while other companies such as Alpha-G and Galecia Associates provide consulting, and some companies, such as IndexData, provide a little of each.

Another common misconception about the "free-as-in-no-cost" nature of OSS is the belief that none of the code is produced through paid development. This misconception has been supported by some of the more woo-woo philosophizing found in works such as Eric Raymond's *The Cathedral and the Bazaar* [2], the first half of which tosses around speculative theorizing about "hacker milieus," "gift cultures" and reputation game analyses to create a myth of the noble OSS developer, contributing code for the good of the community for nothing more than an enhanced reputation. Volunteer " and many OSS projects are thriving communities with leaders, followers, contributors, audiences and reputation systems. The library community, with its strong ethos of sharing and openness, is well-suited for volunteer development.

However, the pragmatic investment of money or sweat equity into OSS development is equally if not more significant than the role of volunteer contributions – particularly in librarianship, where developers are scarce to begin with and usually involved with far too many projects to donate the gift of development time. Companies such as Equinox, Liblime and Media Flex use the proceeds from service or special-project contracts to fund future development in the products they support, while library organizations contribute developer hours to developing services they will need, as is happening with Project Conifer, a consortium of Canadian academic librarians contributing to the development of acquisitions and internationalization for Evergreen.

This blended development model is unique to OSS – and is directly a result of the other ways OSS is "free." Like so many things librarians hold dear – information, books and library buildings themselves – OSS is open, available and visible for all to see. This consonance with librarian values is a philosophical advantage that should not be downplayed.

Leave the Cloak and Daggers at Home

However, this openness is also a significant strategic advantage. Even if OSS were a financial wash compared to proprietary software, or only neutrally consonant with librarian values, there are benefits to OSS that make OSS preferable to proprietary products.

Vendors for proprietary software have a ready excuse for cloak-and-dagger development: if they developed their code where everyone could see it, they would be compromising the source of their revenue. OSS completely

removes that problem from the table. When anyone can see the code, the code itself has no monetary value, so the economic model for OSS is dependent on the services that software companies can deliver – a point made quite well in the second (and far superior) half of *The Cathedral and the Bazaar*. There is no motivation to reduce maintenance support when that is the company's primary revenue stream.

With OSS, it is also much easier for customers to perform due diligence about the products they are selecting – in other words, no more false promises. Many librarians have lived through years of reassurances that the companies they were working with would deliver the next version Real Soon Now, only to learn through terse press releases that the long-promised product would never be materializing. (Some librarians recall being "trained" on Taos, the never-to-emerge vaporware from DRA.) As Raymond also makes clear in the second half of *The Cathedral and the Bazaar*, "when your key business processes are executed by opaque blocks of bits that you can't even see inside (let alone modify), *you have lost control of your business*" [2, p. 152] (Raymond's rather needless emphasis).

Put on Your Big-Girl Britches

There was a time, decades ago, in the early days of library automation, when libraries built the software that drove their systems. Melvyl, NOTIS and LRS share the proud heritage of systems that were designed and written by libraries and for libraries.

It has been suggested that if the Internet had existed in those days, and those early developers had ready access to file-sharing and online community-building, librarians might have invented open source. Instead, companies founded by people with the best of intentions took over fledgling software efforts and struggled to build viable businesses. While some vendors proved scurrilous, and some customers proved exasperating, in most cases the vendor-customer relationships were riddled with endemic and unavoidable "no-fault" problems. Library software vendors found themselves dealing with customers who in a chronically under-funded profession were never able to pay realistic licensing fees, while librarians were dealing with vendors who, unable to earn enough revenue through licensing, scrimped on maintenance and development.

"Learned helplessness" is what Lori Ayre of The Galecia Group calls the outcome of this folie à deux. Ayre writes, "It's ridiculous that libraries are stuck with the systems they've got without options to determine what changes get made or even the access or privileges that would allow them to make the changes for themselves." [3]

"Learned helplessness" has resulted in library automation software that is generally years if not decades behind development found outside LibraryLand with the result that library services are often cramped by the limitations of aged library software that all too often falls short both in features (how many catalogs still do not perform spell-check? How many do not allow patrons to pay fees by credit cards?) as well as in functionality (such as poor reindexing or transaction load capabilities). In an era of budget reductions, the last thing libraries need is software that positions them poorly with their communities.

Ayre has further commented on the stagnation in ILS development. "Library system admins simply stopped asking their ILS vendors for the changes they needed after seeing their requests end up in some future release black hole. Library staff soon learned that "it wasn't possible with their system" and stopped asking system admins for changes. Eventually, people just stopped thinking about how things could be improved. Somehow, we now have to reverse this trend."

Ayre has also shared her experiences trying to get proprietary software to communicate with other vendor products. In this area – interoperability – OSS again presents distinct advantages. This superiority becomes sharply clear in discussions about emerging standards, where vendors for proprietary products can spend years in stalemate, because to become compliant for a particular product means that the software vendor must make its code open and available. Foot-dragging about standards compliance may often have much less to do with the vendor's reluctance to hew to a shared model than to open its code for the world to see.

Debunking the Hype

As discussed earlier in this article, there are numerous strong cases to be made for the adoption of open source software in libraries. But in some circles the hype has far exceeded what any software can deliver. It's

important to underscore that beyond the core characteristic of openness, nothing else can be inferred about OSS. The quality of OSS ranges from superior, industrial-strength, ever-adapting programs such as Firefox, Linux, Apache and PostgreSQL, to what can be charitably called GIAG, or Guy in a Garage software – the poorly documented, badly maintained programs about which the most we can say is that it's "free" (less the many hours lost struggling within its limitations). Some OSS programs have grown huge contributor communities – thousands of developers have contributed to the latest Linux releases. Some have one or a handful of developers involved and will never grow beyond that number.

Quite a few OSS programs exhibit the latest characteristics of good software – reliance on modern programs and service-oriented architecture – but this desirable attribute is in part an accident of the relatively recent history of OSS. The history of software makes it clear that at some point not all that far in the future, all OSS known today will be obsolete. For example, among larger applications, there is an argument to be made that PostgreSQL is replacing MySQL for serious database developers.

So while it is beneficial that so much OSS leaves behind the ratty, tatty code used in some of the more notorious examples of elderly library software, it is still incumbent on the library OSS development community to evolve its products in concert with larger changes in software development outside LibraryLand. An open, collaborative model can facilitate healthy code evolution, but it cannot guarantee it.

Not only that, but OSS occasionally lives up to its stereotypes and even some of the FUD (rumors based on Fear, Uncertainty and Doubt) spread about it.

OSS can be developer-centric, with an emphasis on bare-metal interfaces and text-heavy script configurations that privilege the technically savvy at the expense of those who are inexperienced or who prefer to showcase their mettle through other means than hand-editing lengthy scripts. Software that is more time-consuming or complex for general users than its proprietary counterparts can feel like software of last resort, even when the software has superior characteristics that may endear it to power-users or make it the better choice for high-end needs.

Join these problems with some websites associated with OSS projects – with their overly bright web pages reminiscent of mid-1990s web development, cartoonish software "mascots," confusing and vaguely cultic invitations to "join the community" and grudging invitations to download "unsupported" Windows "binaries" (perhaps with a jab at "Micro$oft") – and hesitation about OSS becomes understandable, particularly for library administrators accustomed to products that are superficially more polished and who direct their appeal at the people writing checks, rather than the developers.

Also, while some OSS is exhaustively documented – MySQL is an example – overall documentation is a pervasive problem. Stuart Yeates from the University of Oxford and a contributor to the Educause blog, Open Source in Higher and Further Education, observes in his article, "Documentation Issues in Open Source," that "[m]any open source projects face significant challenges generating and maintaining high quality, end-user documentation." [4]

Yeates traces these challenges to nine issues, including the fact that documentation is rarely considered "sexy." Yeates' recommendations include requiring documentation as part of the code roll-out and underwriting the cost of professional documentation, which are practices embraced in whole or in part by several library software development projects.

One issue Yeates leaves off the table is that most people vastly underestimate the skill and resources required to write good documentation – or for that matter, to produce good writing at all. In an environment dominated by brilliant, dedicated coders, it can be easy to devalue the humble efforts of right-brainers and the skills they bring to writing, graphics design, project management and information architecture.

OSS may always trend to these limitations; these problems may be ones that are in constant mitigation, but never fully resolved. That said, the sometimes messy, poorly documented, developer-centric world of open source development, for all its foibles (endemic or otherwise), is a healthy improvement on the "learned helplessness" of the last two decades.

Oh Brave New World…

OSS presents important opportunities for libraries – though in most ventures, *opportunity* is also a synonym for *risk*. We can take back

ownership of our future, returning software development to its early glory days, when software development was intimately intertwined with, and helped drive, rapid changes in library services. As this author wrote in a biography of the technology pioneer Anne Lipow, there was a time when librarians envisioned major new services such as document delivery, and developers working right in the same library wrote the code to help make these services happen.

This is the world we want to be in again. It will not always be easy, and there will be a few spectacular failures. But there will also be spectacular successes – and this time, they will happen in the open. ■

Resources Mentioned in the Article

[1] WorldCat. (2008). *WorldCat Developer's Network.* Retrieved October 15, 2008, from http://worldcat.org/devnet/wiki/Main_Page.

[2] Raymond, E.S. (2001). *The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary.* Cambridge, MA: O'Reilly.

[3] Ayre, L. B. (2008, July 10). Ten years of learned helplessness coming to an end. *Mentat* [blog]. Retrieved October 15, 2008, from www.galecia.com/weblog/mt/archives/cat_especially_for_libraries.php.

[4] Yeates, S. (2005-2008). Documentation issues in open source. *OSS Watch.* Retrieved October 30, 2008, from www.oss-watch.ac.uk/resources/documentation.xml.

TOP OF ARTICLE
< PREVIOUS PAGE