CHAPTER 1

What Is Open Source Software?

In recent years there has been a great deal of buzz in the library community about open source software (OSS). Open source integrated library systems such as Evergreen and Koha have gained popularity, as have open source next generation catalog tools such as VuFind and Blacklight. So what exactly is open source software, and what advantages does it present for libraries?

The term *open source software* can be confusing at times. Typically, people associate open source with software that you can download and install for free. This definition focuses solely on the cost aspect. However, while some OSS can be downloaded and installed for no cost, cost really isn't what OSS is about. At its core, OSS is about the freedom to adapt and customize software. For this reason, many people refer to open source software as FLOSS (free/libre open source software) to shift the focus to the freedoms that are inherent with this type of software. Dan Chudnov describes it best in a 2008 *Computers in Libraries* column when paraphrasing the GNU Public license (www.gnu.org/licenses/gpl.html): "FLOSS provides the the freedom to run, study, adapt, improve, and redistribute software."[1]

So what does this mean in practical terms? With OSS, you have access to the raw source code files, so if you wish you can view these files and make changes. You have the ability to correct problems, add new functionality, or improve existing functionality. Any changes you make can be submitted back to the applications community and developers for others to try out and adopt. Our very ability to write a book like this and delve into the inner workings of the applications on the level we do is dependent on FLOSS.

Whether you know it or not, you have probably already made some use of OSS. A significant portion of websites run on OSS; in fact, major companies such as Amazon and Google use OSS. Moreover, if you browse the web with Firefox, check your email on Thunderbird, or blog using WordPress, you are already using OSS.

Principles of the Open Source Community

Open source software is very much about community; without a vibrant community, OSS often fails. This is because OSS is community developed, maintained, and supported. This means that code is often written by many different people throughout the world. While there may be some coordination of efforts on a project between open source developers and a core set of developers, there doesn't have to be. Each member of the software community can contribute in the way he or she sees fit.

This model has two advantages. First, many eyes and minds make for better software. The only way to find bugs in software is to look at the system intensively, which is a time-consuming process. Bugs are more likely to be caught when the process is distributed across a large group of people. (And, because a greater number of people have the ability to fix a bug once it is found, this model also increases the likelihood that the bug will be fixed.) Second, this cooperative development brings the "wisdom of the crowd" to open source projects. By bringing together a diverse group of people with different backgrounds and opinions, you get software that reflects a variety of perspectives and needs—in short, you simply get better software.

Another principle of the open source community is that participation is expected. Not only is it okay for developers to copy, correct, and enhance software, it is expected. In fact, if you want to redistribute or repackage open source code, most open source licenses require that you contribute any changes you make back to the community. Development of code is just one way to participate in an open source community, though. There are many other potential tasks, from reviewing code to answering questions in forums to writing documentation. Any kind of participation helps make the community stronger and more effective.

Costs

While in the open source model there is typically no cost for the software itself, *running* open source software is not without cost. The bulk of the cost shifts from the software itself to the expertise needed to run the software and, potentially, modify it. This is one of the most significant differences between OSS and commercial software. While using any software entails costs for implementation and support, with commercial products the software itself also costs money. By

implementing OSS, libraries remove this cost and are able to invest that money elsewhere. Some have suggested if this money were reinvested in software development, the overall quality of library systems could be improved.

Many organizations argue against implementing OSS because they believe that the cost of supporting it is greater than the cost of a commercial software product. This argument makes the false assumption that commercial software doesn't come with a support cost and defers responsibility for support back to the vendor. Organizations that implement OSS choose to take responsibility for the software they run, but with this responsibility comes flexibility. OSS is not tied to a particular vendor; several vendors could choose to support the same open source product. Libraries thus have the flexibility to change vendors without changing their software. Additionally, they can change or modify the code to meet their individual needs as they see fit.

Strengths and Limitations

Open source software has a myriad of strengths. First, of course, is the cost of the software. Since open software itself typically has no cost, it allows libraries to invest these resources in other areas. Second, because the source code is freely available, libraries can inspect and evaluate the software on a deep level without having to purchase it first. Third, OSS provides libraries with the flexibility to adapt software to their specific needs. An in-house developer can customize OSS to meet the specific needs of the library or fix bugs. Libraries are no longer chained to the development priorities of vendors, which often don't agree with the library's priorities. Fourth, as mentioned previously, OSS is not tied to a single vendor. Therefore, libraries have the ability to change vendors without changing their software.

For all these strengths of OSS, it has its limitations as well. There is almost never a built-in support hotline to call, and the level of support from the community varies greatly from application to application. Some applications have extensive documentation, forums, IRC channels, and other forms of support, while with other applications, the only documentation might be the source code itself. Organizations need to do their homework before implementing an open source application to understand the level of support available.

Licenses

Open source software can be distributed under a variety of licenses. Currently, the most popular license for OSS is the GNU General Public License, also referred to as the GPL. Under the GPL users have the right to use, copy, modify and distribute the software. However, if users choose to change or add to the software and redistribute it, it must be redistributed under the same license as the original software. This provision, which is known as *copyleft*, has helped to give OSS contributors the confidence that their work would remain free for others and not be exploited by individuals or organizations that have no intention of contributing back to the community. Ironically, the GPL allows users to choose whether to charge a fee for redistributing the software, as long as the redistribution adheres to the terms and conditions of the GPL.

In addition to the GPL, there are a variety of other licenses for OSS, including GNU Lesser General Public License, BSD License, Mozilla Public License, MIT License, and Apache License. Some of these licenses are GPL-compatible, which means that applications with these types of licenses can have their code combined with GPL licensed software without creating a conflict. However, some of these licenses are not as strongly copyleft as the GPL license. For example, MIT and BSD-based licenses allow derivative works to be available under terms that are more restrictive than those of the original license. While each of these licenses is slightly different, they are all based on the same basic principles and ideals that users should have the freedom to use, copy, modify, and distribute software.

Open Source Business Models

Open source software has become so popular that several companies have developed business models based on it. The first business model, known as the *dual license* model, is the one used by Movable Type and MySQL. Dual licenses enable commercial entities to make two versions of their software available. One version is open source and typically provided at no cost. A second version with additional functionality and features, often designed as an enterprise solution, is available under a proprietary license. Using this model, organizations can both simultaneously support open source development and have a stable business.

This is the model under which the Movable Type blogging platform operates. Originally the Movable Type blogging platform was OSS, but in 2004, Six Apart, the company that developed Moveable Type, changed its licensing model to be strictly proprietary. This change created a huge backlash within the user community and prompted many Movable Type users to move to the WordPress platform. In response, Six Apart decided to dual license the software, offering both open source and proprietary versions for different user markets.

The second model, known as the software as a service model (SaaS), is the one adopted by companies like LibLime and Equinox. This model differs from the dual license model in that revenue is generated not from selling the software itself but from providing support for the software. In this model, an organization chooses the OSS it wants to use, then finds a vendor to implement and support that software. Currently Equinox specializes in implementing and supporting the Evergreen open source ILS. LibLime provides support for a variety of products, including the Koha open source ILS; MasterKey, a federated search tool; and Kete, a digital library tool. However, LibLime's primary business is centered around supporting the Koha ILS. Outside of libraries, there are companies that specialize in supporting other OSS, such as the Drupal or Joomla CMSs. This support-based model seems quite successful, and libraries considering OSS should examine these types of options for external support.

Applications Discussed in This Book

We used several criteria when selecting the web applications to include here. The first two were ease of installation and server requirements. We wanted applications that were fairly simple to install and had fairly typical server requirements. For this reason, the majority of the applications in the books are LAMP-based (see more on the LAMP stack in the next chapter). The next criterion was the level of use and the robustness of the user community. Because libraries installing OSS have to be self-supporting, we felt it was important to choose applications that were frequently used by libraries and that also had fairly active user and development communities.

Lastly, because of the breadth and variety of open source web applications available, we tried to focus on software that is universal to the tasks performed in nearly every library. For this reason, we have chosen not to cover tools related to digital libraries, special col-

lections, or archives. We do understand that each library has individual needs, so where possible we will present more than one option for libraries to choose from in each category.

Endnotes

1. Daniel Chudnov, "What Librarians Still Don't Know About Open Source Software," *Computers in Libraries* 28, no. 3 (2008): 40, 42–43. Library, Information Science & Technology Abstracts, EBSCOhost (accessed November 10, 2009).

References

Chudnov, D. 2007. The future of FLOSS in libraries. In *Information tomorrow: Reflections on technology and the future of public and academic libraries*, ed. R. S. Gordon. Medford, NJ: Information Today, Inc.