

■ KENNETH E. KENDALL, Feature Editor, School of Business-Camden, Rutgers University

Rear Admiral Grace Murray Hopper, one of the very first programmers on the Mark I computer, was called out of retirement to stabilize the COBOL language. She then went on to direct a project that developed a set of programs and procedures for validating COBOL compilers which became the basis for national and international standards for all modern languages. Those of us who witnessed what she overcame in her project are understandably gun-shy when we hear things like “Linux can be customized and can therefore cater to different markets.” In this month’s column, Sameer Verma tells us not to fear. He discusses a philosophy of programming, called Open Source Software, that may revolutionize the software world. The fact is that much of the ecommerce world we know today runs on open source software. Read this dynamic article and take up the challenge Sameer poses: Begin to conduct basic research in this topic.

Open Source Software— As Good As It Gets

by Sameer Verma, San Francisco State University

It was the summer of 1997. I was hanging out with some of my fellow doctoral students at the Decision Sciences Department at Georgia State University. We all had the million dollar question in the back of our minds—“When will I graduate?” To get away from the prospects of answering that question, we decided to do something different and engaging. One of my colleagues suggested that we look at this interesting piece of software called Linux. So, we spent an entire Saturday afternoon looking into the intricacies of Linux, an operating system we knew little about. I came away with a new-found respect for command-line prompts and the joy that I had seen Linux. We also heard the term “open source” being mentioned several times but not being a programmer, I wasn’t quite sure what it meant.

Fast-forward to 2003. I managed to answer the million dollar question and eventually graduated. I now work as an assistant professor in the information systems group at San Francisco State University. I see Linux around me more often than I had expected. Linux is often considered to be the poster child for a wider movement called “open source.” Simply put, open source is a way for developers and programmers to share source code in a way that encourages redistribution and propagation, while ensuring the integrity of the author’s source code. Open source soft-

ware is becoming more visible, although there are implementations that many people use without realizing it.

One of the most visible implementations of open source software is Google, a very popular search engine, which is powered by more than ten thousand Linux servers worldwide (Hammonds, 2003). Yahoo! is another example, where the front-end services are powered largely by over two thousand FreeBSD (another open source operating system) servers (Landley, 2000). The less visible ones are software such as Apache, an open source Web server, which is used to serve Web pages from over 60 percent of the world’s Web servers (Netcraft, 2003). Chances are that you’ve used it without knowing. Conflicting media reports, shaky benchmark tests, and differing points of view can send a very noisy picture about proprietary vs. open source software. This is also sometimes called the FUD factor—fear, uncertainty, and doubt (Raymond, 2001). To minimize the FUD factor and weed through all this, I said to myself, “Why not look at all these as an MIS researcher?” Here are some thoughts.

SDLC vs. Prototyping

Open source is very interesting from a systems development perspective. Traditional approaches to systems development fol-



Sameer Verma

is an assistant professor of information systems at San Francisco State University. He earned his B.Engg. from Osmania University, India, and Ph.D. from Georgia State University. His main areas of interest include

diffusion and adoption of innovative technologies. His current projects are based on security in Wi-Fi networks and messaging.

<http://verma.sfsu.edu/profile/>

low a cycle of analysis, design, development, implementation, and evaluation. Given its step-by-step nature, where one step leads to the next, this approach is also referred to as the cascade or the waterfall model. Open source software development follows a methodology that is in some ways closer to rapid prototyping. As opposed to a planned development cycle, open source development is done in very rapid, very short cycles often repeated several times over a short span. The approach's motto is "release early, and release often." Some open source projects will release up to four versions of the software in a week.

Open source development cycle, however, differs from rapid prototyping methods. Prototyping is the primary method used in an open source project. There is no grand plan for software development. It is more like several small projects, each designed to solve a limited set of problems. These bits and pieces may then be used together to build something much larger. In his book titled *The Cathedral and the Bazaar*, Eric S. Raymond (2001) points to this difference in style as being somewhat like the plan to build a cathedral versus the gatherings of a bazaar. The building of a cathedral (i.e., traditional software) relies on a grand blueprint. It requires elaborate planning to build, and errors in design can be very expensive. The bazaar model on the other hand relies on several small vendors bringing their wares (small pieces of software) to the bazaar without any particular grand plans to build a cathedral-like application.

Let's take the example of Linux. One finds that any distribution of Linux, commercial or otherwise, is essentially a collection of the Linux kernel, originally created by Linus Torvalds, combined with hundreds of other utilities and applications all created independently of one another. Building a distribution of Linux would be akin to someone who shops at a local bazaar and combines various ingredients into a custom dish that has a unique signature. This flexibility is exactly what is behind the variety that open source brings to the table. Some Linux distributions come across seven CDs (SuSE, 2003), while others fit on a floppy disk (LEAF, 2003). Each distribution uses the same Linux kernel, but caters to a different market.

Software and Natural Selection

While this process of picking your ingredients may appear to be haphazard and unorganized, open source software development follows a process which is closer to natural selection. To take another example, let's say that an existing piece of open source software has an identified bug. Two programmers may take it upon themselves to solve this problem independently. Both solve the problem in their own way and submit their work in the form of a patch to the open source project's software repository. Both patches will become available to the open source community to peer review and test. The patch that turns out to be more useful and applicable will continue to be used while the other one will get neglected and eventually die (i.e., not used anymore). The selection process of picking the most applicable combinations makes this process very much like natural selection.

Then there is the issue of quality assurance (QA). Since no particular company owns an open source project (although companies can sponsor open source projects and use open source project for commercial purposes), it is often argued that open source software does not go through the rigors of a commercial product from a traditional software company. The counter argument for QA is captured very aptly by Eric S. Raymond: "Given a lot of eyeballs, all bugs are shallow" (2001). Since the open source software is open to peer review (often a double-blind peer-review process because many creators and users remain behind the anonymous veil of the Internet) by thousands of programmers worldwide, bugs tend to be identified faster and fixed at a similar pace.

Software License—A Driver for Diffusion?

The licensing model provided by software vendors ensures the safety of intellectual property rights and, of course, a steady revenue stream. In the case of end-users, such licenses are often called EULA, or end user licensing agreements. The open source world has its equivalents. The most popular and widely used license in the open source world is called the general public license or GPL (SourceForge.net, 2003). The GPL is a strange license as compared to a

typical license we see with traditional software. The GPL insists that source code be freely available, and any significant modifications that are done to an existing open source project must be released on a publicly available Internet site (Web, FTP, mailing list, etc.). It appears that the sole purpose of GPL is to promote the growth of open source software by forcing the "openness" issue. Contrary to popular belief though, the GPL does protect the intellectual property rights of the contributions by ensuring that credit is given where credit is due.

Open source software cannot be distributed without giving credit to the contributors. By insisting that the source code and any related modifications be made available, GPL not only protects intellectual property rights, but also ensures the propagation of open source software in a community. Another popular open source license called the Berkeley Software Distribution (BSD) license is more commercial friendly. While the BSD license promotes the nature of open source software, it allows commercial interests to keep their modifications internally protected by a proprietary license. Such an approach fosters open source and encourages companies to develop applications on top of existing open source software. One very good example of such an approach is the MacOS X operating system from Apple. MacOS X is built on top of Darwin, which is based on BSD UNIX. Apple however uses a proprietary user interface called Aqua (Apple, 2003). The BSD license allows Apple to use the power of a Unix-based operating system, while maintaining its proprietary edge in the GUI market.

In all there are 43 licenses that fall under the category of open source (OSI, 2003). Each license varies in its terms and conditions, and is designed to foster open source development, protect intellectual property rights, and ensure the diffusion of open source into the software world.

Business Models

Closely related to the topic of licensing is the idea of business models. If a company decides to use open source software for any part of its operations, it has to pay attention to the business model that would be most applicable. It is often argued that it is impossible to run a business on a collec-

tion of software that can be given away for free. There appear to be a few cases that have defied this logic. A few of the commonly used business model approaches are used either to improve market share, visibility, or even profit. Here is an example. When Netscape Communications was competing against Microsoft in the browser market, they realized that to maintain continued presence even after selling the business, Netscape would have to use an approach that would ensure the visibility of its work beyond its sale to AOL Corporation.

In 1998, Netscape Communications decided to release its browser's source code under an open source license called the Mozilla Public License (Raymond, 2001). As a result, Netscape's legacy and code base continues to live inside AOL Corp. as Netscape, and outside AOL Corporation as Mozilla. In fact, after releasing the last line of browsers in the Netscape name (Netscape 4.78), AOL stopped releasing for quite some time. It took Mozilla almost four years to grow from its proprietary roots into an open source project. The original code was almost completely gutted and replaced by better and faster code. This work resulted in Mozilla 1.0, released in July 2002. Interestingly enough, AOL's latest release of Netscape (version 7.02) is not based on a proprietary version of Netscape's source code but on the open source code base of the project Mozilla (Dotzler, 2003). The approach used by Netscape is called the loss leader model. Some of the other approaches that are popular in the open source arena are support sellers, widget frosting, and accessorizing (Raymond, 2001).

Virtual Communities at Their Best

The virtues of a virtual community are based on the coordination of users across geographical and chronological barriers. E-mail, web, video conferencing and instant messaging are a few technologies that are often mentioned in the case of running a virtual community. The growth of Linux and other open source projects was accelerated largely by collaborative tools available to the community. Most open source

projects rely on e-mail, mailing lists, Web based software repositories, compile farms and Internet relay chat forums to collaborate. The growth of the Internet, the Web and easy access to online applications have helped open source projects grow beyond simply conceptualizing an idea. It is typical to have geographically dispersed collaborators on a project that have never seen or met each other. Yet, they are able to collaborate, and in many cases successfully create the software that is used around the world.

Perhaps one of the best examples of the implementation and use of a virtual community is SourceForge.net. This project hosts over 60,000 open source projects with over 600,000 registered users from all over the world (SourceForge.net, 2003). It is considered to be the world's largest open source software support system. Source Forge manages open source projects by providing them with Web space, documentation repository, mailing list support, bug reporting tools, compile farms, and search engines. Incidentally, the creators of SourceForge (VA Linux) also hold the world record for the largest initial public offering in the history of the stock market (Glasner, 1999).

Conclusion

It appears that the tools are out there along with the ideas and the people that created them. Open source is a revolutionary approach to software development. It appears to be more than just a methodology. It is almost as if open source is an entirely different philosophy, in the software sense. So, I invite you to put on your research gloves and hammer out some hypotheses, because when it comes to software, open source is as good as it gets.

References

- Apple. (2003). Darwin. Retrieved, from the World Wide Web: <http://developer.apple.com/darwin/>
- Dotzler, A. (2003). Getting involved with Mozilla: The people, the tools, the process. In L. U. G. O. Davis (Ed.). Davis, CA: Mozilla.org.

Glasner, J. (1999). VA Linux sets IPO Record. Retrieved, from the World Wide Web: <http://www.wired.com/news/business/0,1367,33009,00.html>

Hammonds, K. (2003). How Google grows...and grows...and grows. *FastCompany*. Retrieved, from the World Wide Web: <http://www.fastcompany.com/magazine/69/google.html>

Landley, R. (2000). Yahoo!'s History and Infrastructure. The Motley Fool. Retrieved, from the World Wide Web: <http://www.fool.com/portfolios/rulemaker/2000/rulemaker000302.htm>

LEAF. (2003). Linux Embedded FireWall. LEAF Project. Retrieved, from the World Wide Web: <http://leaf.sourceforge.net/>

Netcraft. (2003). April 2003 Web Server Survey. Netcraft, Ltd. Retrieved, from the World Wide Web: http://news.netcraft.com/archives/webserver_survey.html

OSI. (2003). The Open Source Initiative: The Approved Licenses. Retrieved, from the World Wide Web: <http://www.opensource.org/licenses/>

Raymond, E. (2001). *The Cathedral and the Bazaar, Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, CA: O'Reilly & Associates.

SourceForge.net. (2003). SourceForge.net is the world's largest Open Source software development web site. Retrieved, 2003, from the World Wide Web: <http://sourceforge.net>

SuSE. (2003). SuSE Linux. SuSE Linux AG. Retrieved, from the World Wide Web: <http://www.suse.com> ■

Kenneth E. Kendall
School of Business-Camden
Rutgers University
Camden, NJ 08102
(856) 225-6586
fax: (856) 424-6157
ken@thekendalls.org
<http://www.thekendalls.org>