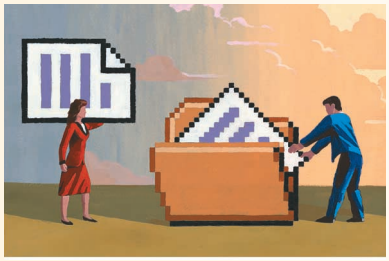
Open Source Software Development: An Overview

Although some challenge the value of open source software development, its popularity cannot be disputed. This overview of open source licensing and development models describes some of the movement's main principles.



Ming-Wei Wu
Ying-Dar Lin
National Chiao Tung University, Taiwan

Proprietary software vendors operate on a closed-source model: They develop their own software and release that software to the public with the intention of gaining market penetration and earning a profit. The open source movement, while still profitable in many ways to profit-oriented companies, relies on a different set of practices. In the open source movement, everyone capable of writing code is welcome to join in, a strategy that—according to open source advocates—directly leads to more robust software and more diverse business models.

While some challenge the general assumptions about the benefits of open source software development,[1] the evidence of popular buy-in cannot be disputed. People everywhere are adopting various open source distributions or participating in the general movement by contributing their own modifications.

We offer an overview of open source licensing and development and strive to clarify some of the main principles underlying the resulting software. Because so much has already been written about open source, we seek only to touch on some of its major themes and provide pointers to essential information about the movement and its general licensing structures.

OPEN SOURCE BACKGROUND

In 1984, Richard Stallman founded the Free Software Foundation (http://www.fsf.org/fsf/fsf.html), a tax-exempt charity that raises funds for work on the GNU Project (http://www.gnu.org/gnu/thegnuproject.html). GNU is a recursive acronym for "GNU's Not Unix" and a homophone for "new." The GNU Project seeks to develop Unix-compatible software and return software to a state of freedom.

Stallman is both an open source evangelist and a major open source contributor as the principal author of the GNU C Compiler (GCC), GNU symbolic debugger (GDB), GNU Emacs, and more. All these packages provide essential tools for GNU/Linux. The Red Hat 7.1 distribution, which collects some 1,016 packages altogether, contains 70 GNU packages.

The purpose of the Free Software Foundation is not to ensure distributing software to the end user without cost, but to ensure that the end user can use the software freely. From the Free Software Foundation's perspective, the term "free software" has nothing to do with price: A program is free software if you have the freedom to run the program, modify it to suit your needs, redistribute copies either gratis or for a fee, and distribute modified versions of the program so that the community can benefit from your improvements.

Because free refers to freedom, not to price, it is not contradictory to say that software can be both for sale and free simultaneously. According to the Free Software Foundation, the freedom to sell copies is crucial: Selling collections of free software on CD-ROM raises funds for free software development. Therefore, according to the open source definition of the term "free," a program that people cannot freely include on these collections does not qualify as free software.

The copyleft and General Public License are designed to guarantee this freedom. Copylefts are, in essence, copyrights with GPL regulations.

Open source software, essentially a superset of free software, exists in almost countless varieties today,

0018-9162/01/$10.00 © 2001 IEEE

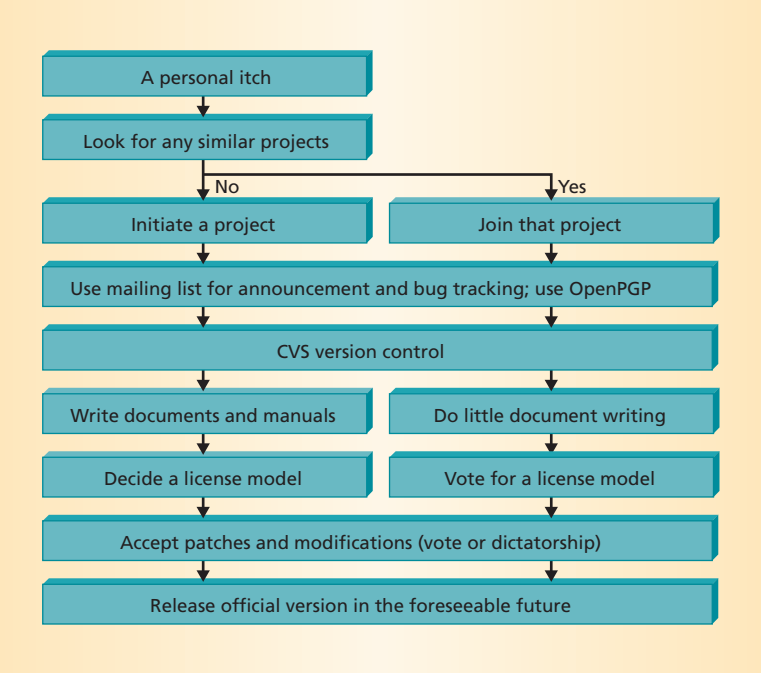


Figure 1. The general open source system development cycle. Allowing multiple participants to contribute to the software development process requires a massive coordination effort.

each with its own unique history.[2] Linux, perhaps the best-known open source software package, began modestly in 1991, seven years after the founding of the Free Software Foundation. Linus Torvalds, at the time a graduate student at Helsinki University in Finland, wrote a Unix-compatible operating system and posted it on the comp.os.minix newsgroup, single-handedly starting the Linux revolution.

Torvalds handed on the kernel maintenance to Alan Cox in 1994 but continued monitoring each kernel version to determine what should be left in and left out. Since 1994, Torvalds has let others deal with user-space issues like libraries, compilers, and the many utilities and applications that go into every Linux distribution. By doing so, Torvalds gives users and vendors the freedom to customize his work.

OPEN SOURCE DEVELOPMENT

The open source software development cycle, as the flow chart in Figure 1 shows, allows literally anyone to participate in the process, but having multiple participants means a massive coordination effort. Developers can use several different models to coordinate these large-scale efforts, from standardizing software—see the "Standardizing Linux" sidebar—to offering participants T-shirts or other benefits. eXtropia uses the open source model to continually acquire contributions from the hundreds of participants who have helped the company produce well-documented, feature-rich Web applications.

Given that project codevelopers may be scattered across the globe, they must agree on a version control system to avoid development chaos. Currently, developers can choose from three major multiple-developer models for version control. Larry Wall's diff and patch for Unix offers one of the oldest standard ways to submit contributions. The diff process discovers the differences between two files to generate patches. This easy-to-use mechanism cannot track a project's history automatically, however.

Walter Tichy's Revision Control System improves the capabilities of diff and patch by automatically keeping track of change history, but his method does not address several important issues. For example, RCS uses the lock-modify-unlock development style that blocks other codevelopers from modifying the code simultaneously. Further, RCS does not support network development, which means codevelopers must work on the same machine.

In 1986, Dick Grune undertook an RCS overhaul that eventually created the Concurrent Versions System,[3] which makes RCS easier to use. Brian Berliner then rewrote CVS using C, and, finally, Jim Kingdon added network development support to the method. CVS uses the copy-modify-merge development style that allows several codevelopers to copy source code from a CVS repository and modify their own versions simultaneously. CVS then intelligently merges all changes together. For example, if version A and B have no conflicts, it's an easy merge; if version A and B have conflicts, the newer version's author must solve these conflicts before merging new contributions into the repository.

Several Web-hosting companies offer free hosting services to support the development of open source projects. For example, SourceForge (http://sourceforge.net) currently offers features such as bug tracking, project management, forum services, mailing list distribution, and more. Both Gnutella and Freenet—Napster-like file sharing systems—use SourceForge for development.

LICENSING MODELS

Source license models fall into three general categories: free—the program can be freely modified and redistributed; copyleft—the owner gives up intellectual property and private licensing; and GPL-compatible—licenses are legally linked to the GPL licensing structure.

In addition to open source licensing models, developers use hundreds of other licensing models for the many kinds of software they market, ranging from shareware to giftware to proprietary agreements, or anything in between. Each of these models contributes to the general confusion surrounding licensing arrangements and the terminology that describes them, because ordinary users seldom read software licenses in detail. Some common open source license models include:

- *General Public License.* This free software licensing uses the copyleft model, a self-perpetuating spiral model that strictly ensures distribution of any derivative work under the same license model.
- *Lesser GPL.* Once known as library GPL, LGPL lets users extend the source with proprietary modules.

- *Berkeley Software Distribution.* The BSD model offers free code distributions and allows covering derivative works under different terms as long as the necessary credit is given. Examples of BSD licensees include Apache, BSD-related OSs, and free versions of Sendmail.
- *Mozilla Public License.* MPL requires distributing derivative works under MPL, which means that derivative work loses patent rights but still can enjoy private licensing. However, a module that MPL covers cannot legally be linked together with a module that GPL covers.
- *Netscape Public License.* This MPL extension permits Netscape to use your added code even in its proprietary versions of the program.
- *Qt Public License.* A noncopyleft free software license, QPL requires distributing any modified source distributions only as patches.
- *Artistic License.* Nearly identical to the GPL model, AL doesn't require distributing derivative works under the same terms when a company uses them internally.

We've listed several of these licensing models in Table 1 for easy cross-reference. For a more complete list of the main open source licensing models with links to their complete information, visit http://www.opensource.org/licenses/.

Among open source licensing structures, although the GPL license calls for the strictest regulation, complaints and public scorn currently provide the main methods for opposing GPL violations. Despite the absence of harsher sanctions, most companies are willing to correct licensing problems and release the modified version of their software to avoid a damaged reputation.

For example, nVidia modified the XFree86 driver for use in its graphics drivers, but did not release the code. Because part of the drivers' code falls under the GPL model, nVidia had to remove all GPL code, then re-release the drivers. In a similar case with a different outcome, Microsoft bought Softway Systems, makers of GPL-regulated software, and repackaged its products as Microsoft Interix to provide a Unix environment within Windows. By doing so, Microsoft could claim Interix as its own work, thereby skirting the GPL regulation.

Many commercial companies have begun using multilicensing models to avoid GPL violations. For example, Sun's StarOffice adapts three licensing models: GPL, LGPL, and SISSL. Ordinary users who can fulfill GPL regulations can use StarOffice under GPL.

Standardizing Linux

The Linux Standard Base seeks to assure cross-distribution and backward compatibility of Linux applications without impeding innovation. Shared libraries are at the root of many application compatibility issues, especially when developers do not subject libraries to strict version control or when application writers don't know which version of a library to use.

Most OSs rely on shared libraries to provide applications with a set of standard functions and utilities that do not waste storage space. Linux, for example, usually includes essential and commonly used libraries such as glibc, pthreads, libm, Xt, and ncurses, among many others.

Applications compiled with a given version of a shared library will expect to find precisely the version they need at runtime. While it is possible for these applications to run with a later version of a library, developers cannot always guarantee this backward compatibility.

One way around this problem is to include multiple versions of libraries within a Linux distribution and allow applications to select the version they were built to use. While this sometimes works, it isn't always practical because adding library versions can use excessive space, undoing the value of having shared libraries. As a result, a system can appear to have several versions of a library, when in reality it only has several links that point to a single file.

While this situation commonly occurs because multiple applications require different minor versions of a given library, it is an impractical strategy for solving library compatibility. First, this approach simply isn't reliable enough. It also does nothing to prevent an installation from overwriting libraries with newer versions that may break backward compatibility. Fortunately, Linux's open source nature makes it nearly impossible for any single provider to make standards a moving target. Users can obtain the latest versions of core Linux libraries, regardless of the Linux distribution they use.

LSB adds one more level of insurance. Regardless of how updates to an LSB-compliant Linux offering occur, they will not break LSB-compliant applications because the LSB libraries will remain untouched.

Adopting LSB standards offers many potential benefits. Well-defined standards provide guidelines that both noncommercial and commercial developers can use to produce good code. Broad LSB compliance means applications will run on every Linux box instead of being limited to a smaller segment of the Linux market. LSB gives every Linux provider access to a larger market because it encourages more suppliers, resellers, developers, and independent software vendors to support Linux. There is still a strong incentive for everyone to innovate without posing a threat to the open source nature of Linux itself.

For more information about LSB, see http://www.linuxbase.org.

Table 1. Open source licensing models.

Licensing model	Free software	Open source	Copyleft	GPL-compatible	Examples
GPL	Yes	Yes	Yes	Yes	CVS
LGPL	Yes	Yes	Partial	Yes	GNU C library
X11	Yes	Yes	No	Yes	XFree86
Python	Yes	Yes	No	Yes	Python
BSD	Yes	Yes	No	No	Apache, Sendmail
MPL/NPL	Yes	Yes	No	No	Mozilla
QPL	Yes	Yes	No	No	Qt
Sun Industry Standard Source License (SISSL)	Yes	Yes	No	No	Commercial-version StarOffice
Artistic License (AL)	No	Yes	No	No	Perl
Apple Public Source License (APSL)	No	Yes	No	No	Darwin

Proprietary developers and companies can use LGPL or SISSL, which both state that the source should be available only when needed to make certain modifications that address issues like incompatibility. Sun's decision to use a multiple licensing model means that software developers and users can make their own choices between freedom and ownership.

BUSINESS MODELS

Although Linux is gaining market share rapidly—see the "Linux Distributions" sidebar for a brief overview of the major packages—most desktop users remain with Microsoft. Those who seek alternatives, however, find a huge library of open source software. These offerings fall into three major categories: operating environments that provide consoles and GUI interfaces, daemons that provide various services, and programming toolkits and libraries that offer development functionality.

Although more proprietary packages are available, there are so many open source packages that most users can find an application that exactly meets their needs. Table 2 lists some of the most popular open source packages, showing their broad availability. Alternative solutions are available for users seeking a shift from, say, Windows to Linux.

Open source software does not cost much even when users purchase it from a third party such as Red Hat. Far more flexible than closed systems, open source software frees both software developers and hardware man-

Linux Distributions

Every open source distribution vendor builds its version around the same evolving kernel. The vendors who publish the various Linux distributions test, integrate, and assemble these packages on top of the kernel. The following list describes a brief sampling of the most popular Linux distributions. For a more complete directory, see http://www.linux.org/dist/index.html.

- *Slackware* (http://www.slackware.com), widely used, somewhat commercial, very stable, and easy to manage, has a long history. You use pkgtool to control its TarBall packages. The rmp2targz tool can convert Red Hat packages for installation in Slackware.
- *Debian* (http://www.debian.org), a distribution that nearly 500 volunteers have formed and maintain, is not designed to turn a profit, but instead to promote frequent interaction between contributors and users. The Debian community gives credit to the appropriate developers in clear detail. Many advanced users find a great deal of flexibility in the Debian distributions.
- *Red Hat* (http://www.redhat.com), perhaps the largest distribution vendor in terms of both sales volume and market share, makes a package that's easy to install, uninstall, and upgrade. It also makes software dependency transparent. Red Hat gives feedback to the open source community and actively recruits open source project developers.
- *SuSE* (http://www.SuSE.com) leads the market in Europe with nearly 30 percent penetration. Known for excellent documentation and abundant package resources, SuSE is a good choice for newbies.
- *Linux-Mandrake* (http://www.linux-mandrake.com), an up-and-coming distribution vendor, began by simply combining a Red Hat distribution with the K Desktop Environment and many other unique, feature-rich tools. This combination proved so popular that its distributors founded MandrakeSoft, which soon became the second most successful Linux vendor.

ufacturers from following a closed-software vendor's specifications. Perhaps best of all, open source software is *reliable*. Perl, sendmail, and Apache have shown significantly more stability than their proprietary counterparts. Because a diverse community of enthusiasts contributes continually to open source development, users capable of programming can quickly solve most problems that afflict the software, such as system bugs, security holes, and even performance tuning.

Private licensing usually takes place when a proprietary software company cannot fulfill GPL regulations. Thus, a private license lets a company keep its code secret. If you are the only owner of a piece of work, you can decide what you want to do about licensing. If, however, you share the work with several contributors, you must re-create the work of each dissenting contributor if you want to sell a private license.

Most Linux vendors do more than simply sell the software. Red Hat, for example, leverages its value-added services, relying not on profits that the software generates but on the revenue from charges for the services the company provides to its vast number of users. For example, Red Hat charges for setting up an Apache Web server, developer training, or 24-hour unlimited tech support for one year.

A company can also gain many related benefits from open source development and distribution, such as enhancing its reputation. For example, GNU has a reputation for releasing well-known coding packages, which makes it much easier for the organization to convince people to use its tools and services. In 1998, Netscape released its source code and rapidly gained market share, becoming the first-choice browser in most Linux distributions. This market position in turn helps the company sell its server products.

While free software does give its users unprecedented flexibility, stability, and freedom of choice, various distributions tend to compete and imitate one another. As the "Standardizing Linux" sidebar describes, the Linux Standard Base organization promotes solutions to these fragmentation problems by facilitating the standardization of the various Linux platforms. The issue of fragmentation will likely

Table 2. Popular open source packages.

Application	Popular packages
Editors	Vim (VIsual editor iMproved), vi, Pico, Joe, Emacs, XEmacs
Word processing	WordPerfect, Kword, Papyrus, Tex/LaTex, LyX, xfig
Office suites	StarOffice, KOffice, iOffice2000
Image manipulation tools	GIMP, XV
Image browsing	Imanager for ImLib, QtVu, Quick Image Viewer, KuickShow
Multimedia	XAnim, XMovie, MPEG TV
MP3	XMMS, X11AMP
ICQ	Licq, Kicq, GnomeICU
Browsers	Netscape Communicator, Opera, Mozilla
E-mail	Fetchmail, mailx, Pine, elm, Balsa, AlphaMail, TWIG, WebMail
Newsgroup clients	Pan, News Peruser, KRN, Tin
FTP	gFTP, nFTP, SkateFTP, IglooFTP PRO, ncftp
File management	Kruiser, Xfm, IInIxdir
Theme	Enlightment, Window Maker, Blackbox, sawfish, Afterstep
Peer-to-peer file sharing	Gnutella, Gnapster, Freenet, Publius
E-mail server	EMUmail, Epop, teapop, Qmail, Sendmail
Newsgroup server	Leafnode, MetaNews
FTP server	BeroFTPD, WuFTPD, ProFTPD
Database	MySQL, PostgreSQL, DBMaker
Web server	Apache, iPlanet Web Server, NetMAX WebServer, Understudy
Development toolkit	GNUPro Toolkit, BXPro, GCC, Code Crusader, Code Fusion
Debugger	DDD, GDB, KGDB
Development platform	Gnome, GNUstep, KDE
Interpreters	Java, Perl, Python, CINT
Palm programming	GCC, prc-tools, PilRC, PocketC

Additional Resources

- Definition of Free Software According to GNU Model, http://www.gnu.org/philosophy/free-sw.html.
- K. Hafner and M. Lyon, *Where Wizards Stay Up Late: The Origins of the Internet*, Simon & Schuster, New York, 1996.
- N. Newman, "The Origins and Future of Open Source Software," http://www.netaction.org/opensrc/future/.
- Open Development Issues, http://www.opendeveloper.org.
- Open Source Developer Network, http://osdn.com.
- Open Source General Directory, http://www.openresources.com.
- Open Source Web Development Resource, http://www.devshed.com.
- E.S. Raymond, "The Cathedral and the Bazaar," http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.
- E.S. Raymond, "The Halloween Documents," http://www.opensource.org/halloween/.
- R. Stallman, "The GNU Project," http://www.gnu.org/gnu/thegnuproject.html.
- M. Stoltz, "The Case for Government Promotion of Open Source Software," http://www.netaction.org/opensrc/oss-report.html.

be the most prominent hurdle that open source software encounters in the years ahead.

Meanwhile, Microsoft prepares for patent warfare—seeking and securing patents that threaten open source. Indeed, last year, Microsoft applied for 25 percent more software patents than it did the year before. The company has hinted at adopting the open source model, but we won't know for certain whether the announcement Steve Ballmer made a few months ago is legitimate until the company releases its source code.

Given the momentum the open source movement enjoys today, it will be interesting to see the extent to which traditional commercial developers will evolve to keep pace with it. Several surprising waves of growth and innovation have swept over the computer industry during the past 30 years, such as the first microcomputers in the 1970s, mass-produced commodity PCs from the mid-1980s through the present, and the rise of the Internet, and it may soon face a deluge of commercial software based upon open source development models. ✷

References

1. N. Bezroukov, "Open Source Software Development as a Special Kind of Academic Research (Critique of Vulgar Raymondism)," http://firstmonday.org/issues/issue4_10/bezroukov/index.html.
2. G. Drummond, "Open Source Software and Documents: A Literature and Online Resource Review," http://www.omar.org/opensource/litreview/.
3. K. Fogel, *Open Source Development with CVS*, The Coriolis Group, Scottsdale, Ariz., 1999, pp. 5-7, 81-99, http://cvsbook.red-bean.com/cvsbook.html.

Ming-Wei Wu *is a graduate-student researcher in the Department of Computer and Information Science at National Chiao Tung University in Taiwan. His research interests include peer-to-peer resource-sharing networks and VoIP integration in PSTN, Internet, and 3G wireless networks. He received a BS in computer science from Soochow University, Taiwan. Contact him at benson@cis.nctu.edu.tw.*

Ying-Dar Lin *is a professor of the Department of Computer and Information Science at National Chiao Tung University in Taiwan. His research interests include design, analysis, and implementation of network protocols and algorithms, wire-speed switching and routing, quality of service, and intranet servers. He received a PhD in computer science from the University of California, Los Angeles. He is a member of the IEEE and the ACM. Contact him at ydlin@cis.nctu.edu.tw.*