

12-26-2001

Open Source Software: A History

David Bretthauer

University of Connecticut, dave.bretthauer@uconn.edu

Follow this and additional works at: http://digitalcommons.uconn.edu/libr_pubs



Part of the [OS and Networks Commons](#)

Recommended Citation

Bretthauer, David, "Open Source Software: A History" (2001). *UConn Libraries Published Works*. Paper 7.
http://digitalcommons.uconn.edu/libr_pubs/7

This Article is brought to you for free and open access by the University of Connecticut Libraries at DigitalCommons@UConn. It has been accepted for inclusion in UConn Libraries Published Works by an authorized administrator of DigitalCommons@UConn. For more information, please contact digitalcommons@uconn.edu.

Open Source Software: A History

by David Bretthauer

Network Services Librarian,

University of Connecticut

Abstract: In the 30 years from 1970-2000, open source software began as an assumption without a name or a clear alternative. It has evolved into a sophisticated movement which has produced some of the most stable and widely used software packages ever produced. This paper traces the evolution of three operating systems: GNU, BSD, and Linux, as well as the communities which have evolved with these systems and some of the commonly-used software packages developed using the open source model. It also discusses some of the major figures in open source software, and defines both “free software” and “open source software.”

Since 1998, the open source software movement has become a revolution in software development.

However, the “revolution” in this rapidly changing field can actually trace its roots back at least 30 years.

Open source software represents a different model of software distribution that with which many are familiar. Typically in the PC era, computer software has been sold only as a finished product, otherwise called a “pre-compiled binary” which is installed on a user’s computer by copying files to appropriate directories or folders. Moving to a new computer platform (Windows to Macintosh, for example) usually requires the purchase of a new license. If the company goes out of business or discontinues support of a product, users of that product have no recourse. Bug fixes are completely dependent on the organization which sells the software. By contrast, open source software is software which is licensed to guarantee free access to the programming behind the pre-compiled binary, otherwise called the “source code”. This allows the user to install the software on a new platform without an additional purchase, to get support (or create a support mechanism) for a product whose creator no longer supports it. Those who are technically inclined can fix bugs themselves rather than waiting for someone else to do so. Generally there is a distribution mechanism, such as anonymous ftp, which allows one to obtain the source code, as well as pre-compiled binaries in some cases. There are also mechanisms for which one may pay a fee to obtain the software as well, such as on a CD-ROM or DVD, which may also include some technical support. A variety of licenses are used to ensure that the source code will remain available, wherever the code is actually used.

To be clear, there are several things Open Source is not—it is not shareware, public domain software, freeware, or software viewers and readers made freely available without access to source code. Shareware, whether or not one registers it and pays the registration fee, typically allows no access to the underlying source code. Unlike freeware and public domain software, open source software is copyrighted and distributed with license terms designed to ensure the source code will always be available. While a fee may

be charged for the software's packaging, distribution or support, the complete package needed to create files is included, not simply a portion needed to view files created elsewhere.

The philosophy of open source is based on a variety of models which sometimes conflict; indeed it often seems there are as many philosophies and models for developing and managing open source software as there are major products. This article will review the development of several major open source projects and attempt to note philosophies of individual projects' creators and maintainers.

The history of open source is closely tied to the history of the hacker culture, since it is largely hackers who have sustained this movement. "Hacker" is used here in the sense of one who is both a skilled professional programmer and a passionate hobbyist wishing to advance computer science, rather than the definition recently used by the popular press of a destructive system cracker. Eric Raymond's essay, "A Brief History of Hackerdom"¹ gives an excellent overview of the development of the hacker culture.

Cultural and Philosophical Expectations

There are cultural norms common to nearly all mature, sustained open source projects. Eric Raymond has discussed and demonstrated them more thoroughly in "Homesteading the Noosphere"² than space allows here, but several bear repeating. Among them:

- despite license terms which allow anyone to revise source code, there is usually one person (or a very small group of volunteers) who maintains control of the software and incorporates patches, bug fixes,

¹ Eric S. Raymond, "A Brief History of Hackerdom," in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (Sebastopol, CA: O'Reilly & Associates, 1999) and <http://tuxedo.org/~esr/writings/cathedral-bazaar/hacker-history/>.

and added features contributed by others as new releases. This person is often the original creator of the software package, or has volunteered to succeed the creator and received the creator's blessing to carry the project forward.

- often an open source project will have a developer's discussion list (which could be a mailing list using any listserv-type software such as mailman, or a usenet newsgroup, etc.), where people who contribute patches, bug fixes, and new features discuss their ideas and issues.
- usually there is a separate discussion list for users of the software, who often are not very technically oriented and who do not normally contribute to the source code, but who can report problems and ask for help, both from other users and any developers or maintainers who monitor that list.
- the project and software have a website dedicated to that project. The package site may or may not have its own domain name (i.e., <http://www.apache.org>)
- maintainers announce new releases of software at such websites as freshmeat (<http://www.freshmeat.net>) and sourceforge (<http://www.sourceforge.org>)
- while documentation is part of a package, widely accepted packages will often have additional written material in books from trade publishers such as O'Reilly and Sam's.

Richard Stallman, GNU, and the Free Software Foundation

Richard Stallman does not identify himself as part of the Open Source software movement, instead preferring the term "Free Software." (an explanation of the differences between open source and free software is detailed below in "Development of the Term "Open Source"). Regardless of this distinction, Stallman is responsible for laying much of the groundwork for what has become the open source movement. He worked as a programmer at the Artificial Intelligence Lab at MIT in the 1970's and early 1980's, using a locally-

² Eric S. Raymond, "Homesteading the Noosphere," in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (Sebastopol, CA: O'Reilly & Associates, 1999) and <http://tuxedo.org/~esr/writings/cathedral-bazaar/hacker-history/>.

developed operating system called ITS, or Incompatible Timesharing System. He describes his work situation:

“I had the good fortune in the 1970’s to be part of a community of programmers who shared software. Now, this community could trace its ancestry essentially back to the beginning of computing. In the 1970’s, though, it was a bit rare for there to be a community where people shared software. And, in fact, this was sort of an extreme case, because in the lab where I worked, the entire operating system was software developed by the people in our community, and we’d share any of it with anybody. Anybody was welcome to come and take a look, and take away a copy, and do whatever he wanted to do. There were no copyright notices on these programs. Cooperation was our way of life. And we were secure in that way of life. We didn’t fight for it. We didn’t have to fight for it. We just lived that way. And, as far as we knew, we would just keep on living that way.”³

When he wanted to improve upon the printer driver for a laser printer with a tendency to jam which Xerox had given MIT, he was unable to because Xerox had not and would not supply a copy of the source code. Furthermore, he could not get a copy of the source code from a colleague at Carnegie Mellon, because that colleague had signed a non-disclosure agreement with Xerox. When Stallman was not permitted to improve upon the software, he later said, “This was my first encounter with a non-disclosure agreement, and I was the victim....non-disclosure agreements have victims. They’re not innocent. They’re not harmless.”⁴

Eventually the computer system used by the Artificial Intelligence Lab was replaced, making all of their lab’s previous coding obsolete, and forcing the lab to use a new, proprietary operating system. Stallman watched the collapse of his programmer community, and began to look for an alternative. That led him to the concept of free software. He decided to create an operating system complete with all necessary software tools, such as editors, compilers, and utilities, and decided it should be UNIX compatible so that programmers could use it

³ Richard M. Stallman, “Transcript of Richard M. Stallman’s speech, ‘Free Software: Freedom and Cooperation,’ New York University in New York, New York, on 29 May 2001” in *GNU’s Not Unix!* web site <<http://www.gnu.org/events/rms-nyu-2001-transcript.txt>> (23 August 2001).

⁴ Stallman, Free Software: Freedom and Cooperation.

without having to learn a new operating system. He settled on GNU (pronounced “guh-NEW” in this case) as a name for this operating system, a recursive acronym for “GNU’s Not UNIX.”

In January 1984, he resigned his position at MIT to begin developing GNU. He resigned so that MIT would not be able to interfere with the distribution of GNU as free software. However, Professor Winston, then head of the MIT AI Lab, invited him continue to use his former office and facilities, and he began to develop the pieces. In early 1985, he released the first piece which other programmers were interested in using, an editor called GNU Emacs. He made it available for free by anonymous FTP, but at that time access to the Internet was not very common. As an alternate means of distributing the software, he offered to send people the package on tape for \$150. Within a few months he was receiving 8-10 orders per month, which allowed him to pay his living expenses. “So, that was fine, but people used to ask me, ‘What do you mean it’s free software if it costs \$150 dollars?’...the reason they asked this was that they were confused by the multiple meanings of the English word ‘free’. One meaning refers to price, and another meaning refers to freedom. When I speak of free software, I’m referring to freedom, not price. So think of free speech, not free beer.”⁵

Stallman defines free software as possessing four essential freedoms:

- You have the freedom to run the program, for any purpose.
- You have the freedom to modify the program to suit your needs. (To make this freedom effective in practice, you must have access to the source code, since making changes in a program without having the source code is exceedingly difficult.)
- You have the freedom to redistribute copies, either gratis or for a fee.
- You have the freedom to distribute modified versions of the program, so that the community can benefit from your improvements.⁶

⁵ Stallman, *Free Software: Freedom and Cooperation*.

⁶ Richard Stallman, “The GNU Operating System and the Free Software Movement,” in *Open Sources: Voices From the Open Source Revolution*, edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, CA: O’Reilly & Associates, 1999), p. 56.

Beyond the concept of freedom, another interesting development was that as the number of programmers using GNU Emacs grew, some of them started to send him messages noting bugs in his source code and offering fixes. Others sent new source code to add new features. Soon these kinds of messages “were pouring in on me so fast that just making use of all this help I was getting was a big job. Microsoft doesn’t have this problem.”⁷

Stallman also made a practice of incorporating source code written by others wherever possible. Often the determining factor for including the work of others was the terms of distribution. For example, this was behind his decision to use the X Window graphical user interface as part of GNU, rather than writing a new windowing system from scratch.

With work on GNU progressing, Stallman needed a way to protect his work from being taken and used in proprietary packages. To ensure this protection, Stallman developed the general concept of copyleft.

Traditionally software is made available either by its author releasing it into the public domain, or by closing the source code and using copyright and licensing terms to protect it so it cannot be modified. Each presented a problem for Stallman: releasing software in the public domain means anyone can take it and appropriate it for their own use, including copyrighting it themselves and licensing it as a proprietary product. Releasing it with restrictive copyright and license terms prevents the entire user review, bug fix, and feature addition mechanism which Stallman had found valuable. “To copyleft a program, we first state that it is copyrighted; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the program’s code or *any program derived from it* but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable. Proprietary software developers use copyright to take away the users’ freedom; we use copyright to guarantee their freedom. That’s why we reverse the name, changing “copyright” into “copyleft.”⁸

The specific method Stallman used to copyleft GNU was a licensing agreement he developed called the GNU General Public License (GNU GPL). The first version was released in 1989; the second and current version was released in 1991.

⁷ Stallman, *Free Software: Freedom and Cooperation*.

⁸ “What is Copyleft?,” in *Free Software Licenses - GNU Project* 15 September 2001 <<http://www.gnu.org/licenses/licenses.html>> 22 October 2001.

To support the development of GNU, Stallman founded the Free Software Foundation founded in October 1985. It is “a tax-exempt charity that raises funds to promote the freedom to share and change software. And in the 1980’s, one of the main things we did with our funds was to hire people to write parts of GNU. And essential programs, such as the shell and the C library were written this way, as well as parts of other programs.”⁹ While the Free Software Foundation accepts donations, “most of its income has always come from sales—of copies of free software, and of other related services¹⁰

By 1991, Stallman and his programmers had written everything for GNU except the kernel, the part that ties the entire system together. By that time, Linus Torvalds had released the Linux kernel, and he and others combined it with the rest of the GNU operating system (see Linus Torvalds and Linux below). Almost invariably this operating system has ever since been referred to as Linux. Stallman argues, and some evidence demonstrates, that it should more correctly be called GNU-Linux.

Stallman has argued that free software is very much in interest of business. It gives control to businesses over what the software does or does not do. Businesses which need additional functions can hire programmers to add features if the required skills are not available in house. In addition, support can be handled the same way. It also ensures privacy and security for the business. “[W]hen a program is proprietary, you can’t even tell what it really does...it might have a backdoor to let the developer get into your machine. It might snoop on what you do and send information back. This is not unusual.”¹¹

According to Stallman the worst threat to the free/open source software community comes from the use of software patents instead of copyright as a means of protecting intellectual property rights. Currently, software patents are held and enforced on the compression algorithms which make GIF and MP3 formats possible. Both are widely-used formats on the Web, but the patents are often invisible to end users who do not need to

⁹ Stallman, *Free Software: Freedom and Cooperation*.

¹⁰ Stallman, “The GNU Operating System,” p. 60.

¹¹ Stallman, *Free Software: Freedom and Cooperation*.

pay license fees to view or listen to GIF and MP3 files. However, software developers are required to pay license fees when developing software which can be used to create these files—even if that software is not distributed for a profit. As a result, a creator of an open source package which produces GIF or MP3 files can be sued (as has been threatened to such authors by patent holders). Therefore, the use of the software patent mechanism effectively prevents the creation of open source software.

BSD

What was once Bell Labs' Unix operating system has evolved, after a convoluted path, to become another presence in the open source world as three different operating system products: NetBSD, FreeBSD, and OpenBSD. While these operating systems are less well known than Linux, their development illustrates how open source can work and influence other projects.

The University of California at Berkeley obtained a copy of Unix from Bell Labs in 1974. Over the following four years, Bell Labs and Berkeley enjoyed a strong collaborative relationship which helped UNIX to flourish.¹² However, by 1977, this collaboration also resulted in two distinct branches of the development tree: the Bell Labs UNIX, and the Berkeley Software Distribution, or BSD. BSD was shared with research universities around the world, provided they first purchased a source license from AT&T and with that obtained the full source code for both AT&T. This model encouraged others to view source code and contribute to it development.

¹² (This evolution is described in careful detail in Marshall Kirk McKusick, "Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable" in *Open Sources: Voices From the Open Source Revolution*, edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, CA: O'Reilly & Associates, 1999), pp 31-46. Much of this section is condensed from that source. Also, a timeline depicting the entire history of AT&T Unix and BSD is available at <ftp://ftp.netbsd.org/pub/NetBSD/NetBSD-current/src/share/mis/bsd-family-tree>).

Bell Labs released its final version of UNIX in 1978; “thereafter all Unix releases from AT&T...were managed by a different group that emphasized stable commercial releases. With the commercialization of Unix, the researchers at Bell Laboratories were no longer able to act as a clearing-house for the ongoing Unix research.”¹³ Nevertheless, the research community continued to develop Unix. As a result, the Berkeley Computer Systems Research Group (CSRG) was formed to replace Bell Labs as an organization which could coordinate and produce research further Unix releases.

In the early 1980’s, the CSRG made several significant additions to Unix. Key among these was the addition of ARPANet protocols (TCP/IP). This implementation of TCP/IP has served as the basis of, and sometimes the direct source for, every implementation of TCP/IP since. Eventually, the Berkeley improvements were incorporated in AT&T Unix, but for several years, TCP/IP was only available using BSD.

The cost of the required AT&T source license was a prohibitive \$50,000¹⁴ for vendors who “wanted to build standalone TCP/IP-based networking products for the PC market...[s]o, they requested that Berkeley break out the networking code and utilities and provide them under licensing terms that did not require an AT&T source license.”¹⁵ Since TCP/IP had never been in the Bell Labs’ source code and was developed entirely by Berkeley and contributors to BSD. In June 1989, the first freely-redistributable source code from Berkeley was released as Networking Release 1.

The licensing terms were liberal. A licensee could release the code modified or unmodified in source or binary form with no accounting or royalties to Berkeley. The only requirements were that the copyright notices in the source file be left intact and that products that incorporated the code indicate in their documentation that the product contained code from the University of California and its contributors. Although Berkeley charged a \$1,000 fee to get a tape, anyone was free to get a copy

¹³ McKusick, *Twenty Years of Berkeley Unix*, pp. 34-35.

¹⁴ Greg Lehey, “The Daemon’s Advocate: Anarchies, monarchies and dictatorships,” in *Daemonnews: Bringing BSD Together*, October 2000 <<http://daemonnews.org/200010/dadvocate.html>>, 11 November 2001.

from anyone who already had received it. Indeed, several large sites put it up for anonymous ftp shortly after it was released. Given that it was so easily available, the CSRG was pleased that several hundred organizations purchased copies, since their fees helped fund further development.¹⁶

One member of CSRG, Keith Bostic, noted the popularity of Networking Release I and raised the possibility of producing an expanded release which would include more BSD code. It was pointed out to him producing such a release would involve replacing hundreds of files originally developed by Bell Labs.

Bostic's approach to solving this problem would become the classic model employed elsewhere: using the ARPANet, he solicited interested programmers to rewrite Unix utilities based on the published descriptions of those utilities. The only compensation these volunteer programmers would receive would be their name listed by that of the utility they rewrote in the list of Berkeley contributors. In less than two years, most of the necessary files had been rewritten.

Bostic and two other CSRG members, Marshall McKusick and Mike Karels, then spent several months reviewing every file in the distribution. Ultimately they determined six kernel files remained with Bell Labs code which could not quickly be rewritten. Rather than take the time to rewrite those files, and to avoid the delay of drafting a new license agreement, CSRG released the BSD source code they had as Networking Release 2, with the same terms as Networking Release 1. Again, several hundred organizations paid \$1000 for copies of the distribution.

Another CSRG member, Bill Jolitz, incorporated his own files with the Networking Release 2 distribution and released 386BSD. By several accounts, it was not a very stable release, but Jolitz was particular about the direction of 386BSD to the extent that he apparently alienated many 386BSD enthusiasts. Nevertheless, he took the step of making the source code available via anonymous FTP. When he did not make a practice

¹⁵ McKusick, *Twenty Years, of Berkeley Unix* p. 40.

¹⁶ McKusick, *Twenty Years of Berkeley Unix*, p. 41.

of incorporating contributed bug fixes or producing alternative fixes, a number of 386BSD users formed the NetBSD Group to coordinate the development of this system, and their work became known as NetBSD. This group has always focused on making BSD run on “a large number of hardware platforms.”¹⁷ “The NetBSD Group, in the tradition of CSRG, has also focused on experimentation and research rather than developing the most stable BSD possible. More information is available at <http://www.netbsd.org>.

Another group initially focused continued development exclusively on the Intel x86 platform (it now supports Alpha hardware as well); a few months later this group adopted the name the FreeBSD Project. FreeBSD has become the most widely used BSD, as its managers have focused on inexpensive CD-ROM distribution and ease of installation. The distribution also includes a Linux emulation package which allows Linux programs to run on FreeBSD machines, and access to thousands of open source packages through the “Ports Collection,” which allow relatively simple compiling and installation of these packages from source code. More information is available at <http://www.freebsd.org>.

In 1994-1995, a series of disagreements led to the split of OpenBSD from NetBSD. Theo de Raadt has led development of OpenBSD, focusing on stable, secure distributions which incorporate cryptography. More information is available at <http://www.openbsd.org>.

At the same time 386BSD was being developed in 1991-1992, Berkeley Software Design, Inc. (BSDI) began to market a commercially-supported version (originally called BSD/386, now called BSD/OS) by writing their own replacement kernel files as Jolitz had. AT&T’s subsidiary, Unix System Laboratories, filed a lawsuit to stop BSDI from marketing a product which was implied to be Unix. Eventually, Unix System Laboratories was sold to Novell. In January 1994, the case was settled out of court, and one resolution was that distribution of the source code of the latest version of BSD, called 4.4BSD-Lite, was allowed. Replacing Network Release 2 files with 4.4 BSD Lite involved another major rewrite of all BSD operating systems.¹⁸

¹⁷ “About NetBSD,” *NetBSD* <<http://www.netbsd.org/Misc/about.html>> 2 September 2001.

¹⁸ Jordan Hubbard, “A Brief History of FreeBSD,” in *FreeBSD Handbook* <<http://www.freebsd.org/doc/en-US.ISO8859-1/books/handbook/history.html>> 10 November 2001.

Another result is BSD cannot legally be called Unix, since Unix is now a registered trademark of The Open Group. However, at least one writer bluntly states, “[h]istorically and technically, it has greater rights than UNIX System V to be called UNIX.”¹⁹ .

A significant difference between the BSD license and the GNU GPL is that the BSD license does nothing to prevent the creation of proprietary software packages based on modified BSD code. In fact, Microsoft has repeatedly used FreeBSD code implementing TCP/IP in several versions of Windows, and admitted to doing so even as it was criticizing open source software in June 2001.²⁰ According to Jordan Hubbard, FreeBSD project cofounder, “Rather than take the approach that corporations and other interested parties should be ‘forced’ into cooperating with the Open Source movement, the many commercial and non-commercial software developers who are behind the BSD movement...want their software used by anyone and everyone,” with a concern that the GNU GPL will prevent commercial developers from participating at all in open source.²¹

Linus Torvalds and Linux

In October 1991 an undergraduate student at the University of Finland named Linus Torvalds released Linux kernel version 0.02. In announcing its release, he posted this message to the comp.os.minix newsgroup:

Do you pine for the nice days of Minix-1.1, when men were men and wrote their own device drivers?
Are you without a nice project and just dying to cut your teeth on an OS you can try to modify for
your own needs? Are you finding it frustrating when everything works on Minix? No more all-
nighters to get a nifty program working? Then this post might just be for you.

¹⁹ Greg Lehey, Introduction, in *The Complete FreeBSD*, 3d ed. (Walnut Creek, CA: Walnut Creek CDRom, 1999), p. xxix.

²⁰ Lee Gomes, “E-Business: Microsoft Uses Free Code,” *The Wall Street Journal*, 18 June 2001.

²¹ Jordan Hubbard, “Mister, How Far is Licensing from Utopia?” *Open*, 2.6 (June 2001), p. 48.

As I mentioned a month ago, I'm working on a free version of a Minix-lookalike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02...but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it.²²

Even readers who do not understand the technical terminology can recognize the attitude of wanting to take control of a software project, even at the risk of failure, and the joy of working at an operating system just for the sake of working at it in the hacker tradition.

At the time Torvalds was working on this, BSD source code was not quite fully available, and the GNU kernel, HURD, was mired in development which would eventually take years. But rather than write a complete operating system, Torvalds was already integrating GNU tools with his kernel. It is on this basis that Richard Stallman has argued that Linux should more properly be called GNU-Linux. Marshall Kirk McKusick has pointed out, though, “about half of the utilities that [Linux] comes packaged with are drawn from the BSD distribution.”²³

Torvalds focuses on the technology and the community which built it:

Linux today has millions of users, thousands of developers, and a growing market. It is used in embedded systems; it is used to control robotic devices; it has flown on the space shuttle. I'd like to say that I knew this would happen, that it's all part of the plan for world domination. But honestly this has all taken me a bit by surprise. I was much more aware of the transition from one Linux user to one hundred Linux users than the transition from one hundred to one million users.²⁴

This last statement bears emphasis and further examination, as it belies a development style somewhat different from those of GNU, BSD and Apache. These packages were developed “in a carefully coordinated

²² Linus Torvalds, in “Linux History,” in *Linux International*, ©2001 <<http://www.li.org/linuxhistory.php>> 10 November 2001. Torvalds's entire series of announcements is available at this site.

²³ McKusick, *Twenty Years of Berkeley Unix*, p. 46

²⁴ Linus Torvalds, “The Linux Edge,” in *Open Sources: Voices From the Open Source Revolution*, edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, CA: O'Reilly & Associates, 1999), 101.

way by a small, tightly-knit group of people....[while Linux, by comparison], was rather casually hacked on by huge numbers of volunteers coordinating over the Internet.”²⁵ Raymond describes these approaches to development as “Cathedral” and “Bazaar”, respectively. “Quality [in the “Bazaar” model] was maintained not by rigid standards or autocracy but by the naively simple strategy of releasing every week and getting feedback from hundreds of users within days, creating a sort of rapid Darwinian selection on the mutations introduced by developers. To the amazement of almost everyone, this worked quite well.”²⁶

Eric Raymond further asserts, “[b]y late 1993, Linux could compete on stability with many commercial Unixes, and it hosted vastly more software. It was even beginning to attract ports of commercial applications software.”²⁷ This trend has only continued:

Linux is a project that was conceived some five years after Microsoft began development of Windows NT. Microsoft has spent tens of thousands of man-hours and millions of dollars on the development of Windows NT. Yet today Linux is considered a competitive alternative to NT as a PC based server system, an alternative that major middleware and backend software is being ported to by Oracle, IBM, and other major providers of enterprise software. The Open Source development model has produced a piece of software that would otherwise require the might and resources of someone like Microsoft to create.²⁸

Over the past 5 years, the computer trade press has greatly increased coverage of open source software, primarily Linux. The question for many IT professionals has changed. Instead of asking themselves if they will use open source software in their shops, they are now asking where they will use it.

Eric Raymond’s collection of essays published in *The Cathedral and the Bazaar* has examined the reasons for Linux’s success in great depth. In fact, Raymond deliberately imitated Torvalds’s development style as an

²⁵ Raymond, *A Brief History of Hackerdom*, p. 24.

²⁶ Raymond, *A Brief History of Hackerdom*, p. 24.

²⁷ Raymond, *A Brief History of Hackerdom*, p. 24.

²⁸ Chris DiBona, Sam Ockman, and Mark Stone, “Introduction,” in *Open Sources: Voices From the Open Source Revolution*, edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, CA: O’Reilly & Associates, 1999), p. 17.

experiment when managing the fetchmail project²⁹ and analyzes it in the essay of the same name. A frequently-quoted message from that essay is, “Given enough eyeballs, all bugs are shallow.”³⁰ Torvalds extends this by again focusing on the Linux community: “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.”³¹

Beyond the stability of the project, Linux has also prompted the development of a business model which might not seem possible: selling freely available software, and making a profit. By 1994, a number of distributions of Linux were available for under \$30. These included Yggdrasil, Slackware, Debian, Suse, and others. One, Red Hat Software, Inc., has grown into a publicly traded company. According to Robert Young, CEO of Red Hat, “you make money in free software exactly the same way you do it in proprietary software: by building a great product, marketing it with skill and imagination, looking after your customers, and thereby building a brand that stands for quality and customer service.”³²

Other Widely-Used Open Source Packages

Thus far this article has focused on the development of open source operating systems and related tools. But in fact a number of software packages have been developed as open source projects. Among them:

- In 1987, Larry Wall released PERL 1.0 scripting/programming language. Its current release is version 5.6.

²⁹ Eric S. Raymond, *The fetchmail Home Page*, 08 November 2001, <<http://tuxedo.org/~esr/fetchmail/>> 11 November 2001.

³⁰ Eric Raymond, “The Cathedral and the Bazaar,” in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (Sebastopol, CA: O’Reilly & Associates, 1999), p. 41.

³¹ Torvalds, *The Linux Edge*, p. 109.

- In 1990, Guido van Rossum released Python programming language
- In 1994, Rasmus Lerdorf released PHP/FI web scripting/programming language. Its current release is PHP 4.0.6.
- In 1995, the Apache web server program was released and quickly became the most widely used web server product (which it remains today)
- In the mid-1990's, mSQL, MySQL, and PostgreSQL relational databases were released
- Also in the mid-1990's Andrew Tridgell released Samba, a set of utilities which allows Unix machines to use the same network communication protocol as Microsoft Windows

The developers of these packages have focused on the cultures surrounding the projects as well as the projects themselves. Larry Wall has summed up this emphasis by saying, “As a linguist, I understood that a language without a culture is dead. If you get the culture right, the technology will happen.”³³

While many of these packages originally ran exclusively on Unix, most have been ported to other operating systems, including Windows. The other significant development was the involvement of commercial interests in the open source movement. As already stated, corporations such as IBM and Oracle have ported software to the Linux platform. Just as significantly, commercial training, certification, and support are available for Apache, MySQL, and PostgreSQL, just as they are for Linux.

In January 1998, Netscape released the source code for its browser under an open source license, beginning the Mozilla project. As of this writing (November 2001) Mozilla has not yet produced a “release” version 1.0, but is at 0.9.4. This might seem to classify Mozilla as less than successful, but two issues argue against that idea:

³² Robert Young, “Giving It Away: How Red Hat Software Stumbled across a New Economic Model and Helped Improve an Industry,” in *Open Sources: Voices From the Open Source Revolution*, edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, CA: O’Reilly & Associates, 1999), p. 114

- While development has been relatively slow, it has been steady and recent versions have been quite stable.
- The less visible issue in 1998 was that Netscape made most of its money selling server software. “For Netscape the issue was less about browser-related income (never more than a small fraction of their revenues) than maintaining a safe space for their much more valuable server business. If Microsoft’s Internet Explorer achieved market dominance, Microsoft would be able to bend the Web’s protocols away from open standards and into proprietary channels that only *Microsoft’s* servers would be able to service.”³⁴

Development of the Term “Open Source”

Considering that what is now characterized as the Open Source Movement has been in conscious development for nearly two decades, the term “Open Source” itself has been a relative latecomer. In fact, the term was proposed and voted on by a group of people interested in spreading awareness of the sophisticated tools which had been developed outside the proprietary software development model, who were meeting on a regular basis in later 1997 and early 1998. The term was proposed by Christine Peterson of the Foresight Institute. Christine tells the story:

It was a very deliberate effort at a name change. In late 1997 or very early 1998, a number of us felt the name “free software” was holding back the budding industry/movement. Newcomers always thought “free” meant free-as-in-beer, not free-as-in-speech. Various ideas were kicked around, including at a meeting at Foresight, but none were catching on.

³³ Larry Wall, “The History of PERL,” (presentation delivered at 120th Annual Meeting of the American Library Association, San Francisco, CA, 17 June 2001) as part of the program, *Web Tools and Digital Resources: Open Source Then and Now*

³⁴ Eric S. Raymond, “The Revenge of the Hackers,” in *The Cathedral and the Bazaar*, p.202.

On my own I thought of open source, ran it by my PR friend (who didn't like it) and a couple of personal friends (who did).

At the next meeting of the little group convened by Eric Raymond, this time at VA, I was shy about suggesting the new term—I had no standing with this group. So Todd Andersen, with whom I'd planned the name change effort, just used it casually, not suggesting it as an explicit new term.

It caught on immediately at the meeting, without the others noticing except Todd and me, who winked at each other.

At the end of the meeting, it was pointed out by Todd or me that this new term seemed to be working, and people seemed willing to give it a try.

The others later weren't sure who came up with it, but Todd made sure I got credit, which I usually don't push for. Nice of him. And Eric Raymond has also been good about making sure I get credit for it.

Now I try to go around renaming things all the time. Got a big head from this one success. ;^)³⁵

Eric Raymond remembers the meeting a bit differently: “I remember hearing Christine say ‘open source’ and thinking ‘we have a winner’. It may have slipped by other people, but it didn't get by me. :-)”³⁶

This group registered the domain name opensource.org, defined “Open Source”, developed and OSI Certification, and created a list of licenses which meet the standards for open source certification.

The basic definition is:

³⁵ Christine Peterson, “RE: Quick question about the term ‘Open Source’,” 31 August 2001, personal email (31 August 2001).

³⁶ Eric Raymond, “RE: Quick question about the term ‘Open Source’,” 1 September 2001, personal email (1 September 2001).

- The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources
- The program must include source code, and must allow distribution in source code as well as compiled form.
- The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
- The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time.
- The license must not discriminate against any person or group of persons.
- The license must not restrict anyone from making use of the program in a specific field of endeavor
- The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
- The License Must Not Be Specific to a Product
- The License Must Not Contaminate Other Software by placing restrictions on any software distributed along with the licensed software³⁷

“The Open Source Definition allows greater liberties with licensing than the GPL does. In particular, the Open Source Definition allows greater promiscuity when mixing proprietary and open-source software.”³⁸

This is Richard Stallman’s objection to open source software—that it allows the inclusion of proprietary software and ignores the philosophical issue of software freedom. Without these freedoms, there is no philosophical imperative to improve one’s community. Nevertheless, “[w]e disagree on the basic principles,

³⁷ “The Open Source Definition” Version 1.8, in *OpenSource.Org*, <<http://opensource.org/docs/definition.html>>, 1 September 2001.

³⁸ Chris DiBona, Sam Ockman, and Mark Stone, “Introduction,” in *Open Sources: Voices From the Open Source Revolution*, edited by Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, CA: O’Reilly & Associates, 1999), p.. 3

but agree more or less on the practical recommendations. So we can and do work together on many specific projects. We don't think of the Open Source movement as the enemy."³⁹

This is a point on which many who are active in various competing open source and free software packages reiterate often. While this article has focused on a number of differences between operating systems, approaches to collaboration, and the evolution of various license agreements, this focus is at the micro level. At the macro level, nearly everyone mentioned in this article would prefer a competing open source or free package to a proprietary software package. In the future those who have blazed new trails will continue to argue the finer distinctions between their respective works. However, the various groups involved are willing to work with and support one another's right to choose a different approach to solving a problem. And it is clear these individuals look forward to another generation building upon the successes of the past 30 years.

³⁹ "Why 'Free Software' is better than 'Open Source'", in GNU Project – Free Software Foundation, 20 August 2001 <<http://www.gnu.org/philosophy/free-software-for-freedom.html>> 22 October 2001.