LESSONS FROM OPEN-SOURCE SOFTWARE DEVELOPMENT

Tim O'Reilly

Open source is a term that has recently gained currency as a way to describe the tradition of open standards, shared source code, and collaborative development behind software such as the Linux and FreeBSD operating systems, the Apache Web server, the Perl, Tcl, and Python languages, and much of the Internet infrastructure, including Bind (the Berkeley Internet Name Daemon servers that run the Domain Name System), the Sendmail mail server, and many other programs. The Open Source campaign became international news in 1998 when Netscape decided to make the next version of its Web browser (Mozilla) an

TERRY MIURA

open-source product, and when IBM adopted the Apache Web server as the core of its WebSphere product line.

Officially, open source (which is a trademark of the Open Source Initiative—see www.opensource.org) means more than that source code is available. The source must be available for redistribution without restriction and without charge, and the license must permit the creation of modifications and derivative works, and must allow those derivatives to be redistributed under the same terms as the original work. Licenses that conform with the Open Source Definition include the GNU Public License (GPL), the BSD license used with Berkeley Unix derivatives, the X Consortium license for the X Window System, and the Mozilla Public License.

But open source is more than just a matter of licenses. Some of the most significant advances in computing, advances that are significantly shaping our economy and our future, are the product of a little-understood "hacker culture." It is essential to understand this culture and how it produces such innovative, high-quality software. What's more, companies large and small are struggling to understand how the ethic of free source code distribution affects the economic models underlying their present businesses.

This collection of commentaries attempts to provide some perspectives on open-source software that can be applied by anyone developing software, not just by people fully committed to the open-source model. I've tried to put open source in the broader context of collaborative software development, while four open-source pioneers, Linus Torvalds (Linux), Larry Wall (Perl, patch, rn), Roy Fielding (Apache), and John Ousterhout (Tcl/tk), provide background on their own work.

It's important not to reduce the open-source debate to a question of NT versus Linux, or Microsoft versus the rest of the world. We need to understand how the lessons of open source can be applied to software development across the board. Can companies create better products by giving their user communities more influence in product development and evolution? Can users get more "bang for the buck" from their own custom software by sharing it, and receiving the work of others in return? And as the Internet, rather than the desktop, becomes the focus for new applications, do current open-source projects teach principles of large-scale collaborative work that can be applied fruitfully to endeavors other than software development?

Those interested in the topic of open source are encouraged to read Eric Raymond's seminal paper, "The Cathedral and the Bazaar" (www.ccil.org/~esr/writings), which inspired Netscape's decision to release its browser as open-source software. See also Mark Stone's "Science of the New Renaissance" (www.edventure.com/release1/1198.html), which argues that open source is a natural extension of the Western scientific tradition.

Software: Product or Service?

With Microsoft's rise to dominance in the software industry, it's easy to think of software primarily as a product, something that is developed, packaged, and sold. In fact, shrinkwrapped PC applications represent only a small fraction of the total software in use.

Most large business applications are either internally developed or else so heavily customized that they might as well be. Most scientific applications are "one-off," or if built on top of off-the-shelf tools, include a large custom component. Administration, whether of a network, a large computer system, or a Web site, requires the constant development of small tools, scripts, and "glue applications" to make everything work together. Even in the desktop productivity environment, power users develop macros and other "programs" to automate repetitive tasks.

Whether or not open source is a superior methodology for the development of packaged end-user software products is currently being tested in the marketplace. It is already clear, though, that open source is a superior methodology for the development of this kind of custom software. Giving users of a product access to its source code and the right to create derivative works allows them to help themselves, and encourages natural product evolution as well as preplanned product design.

What's more, whether or not open-source technologies succeed as packaged products, their greatest economic contribution may be the new services that they enable.

The Internet is the most striking example of a marketplace, full of economic opportunity, that has grown out of the open-source software community. The Internet (and the various networks that eventually became amalgamated into it) was both the mechanism for and the product of an enormous explosion of collaborative development. Even now, open-source programs such as Bind, Sendmail, and Apache, plus commercial programs emulating functionality originally developed by the open-source community (Web browsing, Internet email), are the heart of the Internet. Companies ranging from Uunet (and the entire commercial ISP market), Yahoo (and the whole commercial Web), plus suppliers from hardware companies to advertising agencies have benefited from the open-source revolution. The commercial impact can

already be measured in billions of dollars.

Perhaps even more importantly, the open-source process reflects a powerful global trend toward networked collaboration. Just as the printing press enabled the spread of knowledge during the Renaissance period, the Internet is enabling large-scale cooperative development efforts today. The speed at which information spreads has increased by orders of magnitude. What's more, the principles of freedom of speech and information interchange take on new significance in the context of our interaction with computers. After all, what is a program but a form of speech with a computer? And what is open source then but the open, public discourse that has always led to the advancement of human knowledge?

And with the spread of the Web, which mixes human-readable text (HTML files) with programmed functionality in the form of CGI scripts, inline JavaScript, server-side Java, or ASP, the boundaries of human-computer communication and human-human communication are increasingly blurred. The vibrancy of the computer industry may, in the end, be a reflection of the vibrancy of our communication with each other.

Giving users of a product access to its source code encourages natural product evolution as well as preplanned product design.

Innovation and Evolution

One of the problems of commercial product development is that the focus is on creating products that will make a profit for their creators. There are many products that meet real needs that are either too specialized or too early to market to produce the return on investment that either a large company or a venture-backed startup demands.

In fact, many open-source projects have been started to solve a user's particular problem. The return on investment is the solution to the problem. But what the early open-source developers realized was that by giving away their work to the networked community of like-minded developers, they might get an additional dividend of functionality returned by other users of the product.

It was the coincident evolution of wide-area computer networking that gave the real impetus to open source. Previously, user communities were centered around vendors and their products. But the possibility of software distribution via FTP, email, and Usenet news enabled a host of independent developers to create unintended benefits simply by passing software around, in a game of "telephone" that sometimes led to Babel, but more often led to evolutionary advances.

A brief look at the early history of many open-source projects illustrates the way they were developed to solve a small problem that later turned out to be significant.

Eric Allman originally wrote a precursor to Sendmail because it was easier to route email for other researchers at UC Berkeley than it was to give them a login on his machine. No one imagined that 20 years later, email forwarding across heterogenous networks and systems would be a critical Internet application. Larry Wall originally wrote Perl to solve some nagging problems in system administration. Tim Berners-Lee created the World-Wide Web to help high-energy physicists (a tiny market by any measure) share their work. This was in the grand tradition of much of the foundational Internet software, which was developed to enable academic information sharing. Richard Stallman reportedly started on the path that lead to the Free Software Foundation, the GNU project, and ultimately, large parts of Linux, because a vendor would no longer provide source code for a printer driver that was malfunctioning. None of these tremendously influential projects would have started in a traditional economic marketplace, and each of them took years to exert its transforming effect. It is precisely because open source gives individuals the power to attack small problems that it is able to create unexpected innovations.

As Peter Schwartz says in his book, *The Art of the Long View*, (Doubleday, 1991), "People and organizations often organize knowledge concentrically, with the most cherished, vital beliefs at the protected center. At the outer edge are the ideas which the majority rejects. A little closer to the center are the fringes—areas not yet legitimized but not utterly rejected by the center either. Innovation is the center's weakness. The structure, the power, and the institutional inertia all tend to inhibit innovative thinkers and drive them to the fringes."

To be sure, there are significant innovations that

come from a concerted attack on known problems and market opportunities. The mechanisms for undertaking such development projects are well-established. But open source is a low-cost way of increasing the opportunity for surprise.

Open source is a way of life, not a better way of picking potential winners and losers. Open-source programs such as Linux, Apache, and Perl are no longer fringe products—they are well on their way to the center. To generate the next big open-source surprise, you need to provide the conditions of innovation: open standards and protocols that allow people to connect new software easily to existing software, good documentation, and a well-defined extension mechanism that allows people to "roll their own" without having to get permission of a central architect. And of course, you need licenses that allow people to build on the work of others without first getting permission.

Cooperative Development

Many of the assertions of "The Cathedral and the Bazaar," including the now famous "given enough eyes, all bugs are shallow," are based on the idea that the Internet enables a larger community of developers than can be applied to a project by even the largest companies.

By making the users of a product into codevelopers, you speed debugging, improve quality, and gain specialized new features that may eventually turn out to be important to a wider audience. Because open source allows users to "scratch their own itch," features can be introduced with low overhead, and live or die in a marketplace that is much more fecund than even the fevered pace of Silicon Valley venture capitalists (and orders of magnitude more so than the centralized product planning of large companies).

This community development aspect of open source means that user communities, not the products themselves, may be the key determinants of a project's success. As an entrepreneur and angel investor, I am approached by many developers looking to turn a marginal product with little market acceptance into a star using open source as some kind of magic bullet. But it doesn't work that way. The product must meet a real need, and attract a passionate core of users who want even more from it than the original developer has provided. Nor can a vendor with an established product hope that by contributing that product to an existing open-source community, they will suddenly galvanize support for that product. Instead, a product must be open sourced to the community of its users.

This leads to an idea that may be anathema to some open-source advocates because it doesn't adhere to the Open Source Definition, but that may have some merit, especially for large companies wanting to leverage some of the principles behind open source without committing to full free software redistribution, most notably by potential competitors. And that is the creation of what you might call, somewhat facetiously (as did a participant at a recent large company executive briefing on open source), a "gated open-source community."

A company with a large user base might want to keep strict control over who has access to its source code (paying customers), but provide full source code, complete documentation, and mechanisms for customer extensions to be folded back into the core source tree. The company may want to study how open-source projects like Apache manage collaborative development, and apply those principles to interactions with its customers, without planning to release the software to the world. This was in fact the way software was handled in the early mainframe days—software came free with the hardware, and such IBM products as CICS were originally developed by customers and contributed to IBM for maintenance and further development.

The lessons of the Apache project, as outlined by Roy Fielding in his contribution to this section, are extremely relevant in this regard. A community needs to develop processes for voting on new features, deciding who has access to the source tree, and communicating in ways that do not stifle the free-floating development that is so central to the appeal of open source.

Of course, one key question in such an experiment would be the size of the developer community required to get the open source "network effect." Many proprietary developer communities may just be too small. In order to leverage the entire Internet-accessible developer community without giving away the store, vendors are starting to experiment with new licenses. One approach is to distribute source code, and allow unlimited modification and redistribution for noncommercial use, but require a different license for commercial distribution.

This is the approach taken by Sun's recent Java Community Source License as well as by the Aladdin Free Public License used by Peter Deutsch, developer of the widely used GhostScript package. In an interview (www.devlinux.org/ghost/interview.html) Deutsch explained his thinking on "community development":

If you are willing to play by what I think are the 1960s rules, then the Aladdin license gives you exactly the same rights and benefits as the GPL: [the software is] free to use, it's free to copy, and you are free to modify it.

In a nutshell, I see the 1960s rules, or the cooperative

rules, this way: "everybody contributes, so everybody benefits."

Unlike the GPL, I make a very solid distinction between distribution as part of a commercial endeavor and distribution not as part of a commercial endeavor....

The philosophical weight of this is that if you want to play by cooperative rules, you get the benefits of Aladdin's work within the context of those rules. If you are not playing by the cooperative rules, then it's going to cost you something to have the rights to get the value from Aladdin software.

In many ways, the heart of many open-source communities is a social compact between a software developer and his or her users, in which both agree to cooperate under certain rules that are beneficial to both parties. At best, this is also true of developer communities that evolve around proprietary products; the point is that open source substitutes cooperation for financial transactions as the glue that ties the community together.

The Importance of Extensibility

Licenses that lower the barriers to cooperation are a key part of the open-source phenomenon. However, as both Linus Torvalds and Larry Wall argue, part of the success of their creations has also come from technical decisions that have made it easier for others to contribute. According to Linus Torvalds, Linux has succeeded at least in part because it followed good design principles, which allowed it to be extended in ways that he didn't envision when he started work on the kernel. Similarly, Larry Wall explains how he created Perl in such a way that its feature-set could evolve naturally, as human languages evolve, in response to the needs of its users.

In fact, many successful open-source projects have a modular architecture, which allows users to extend the system's functionality without having to change existing core functionality. This allows an open-source project to scale with its community while allowing an original visionary developer (or team) to retain control over the core product.

Obviously, one of the areas for study is where the ideal boundary ought to be between a core product controlled by a single individual or small team, and the input of the user community.

Commercial Product Development on an Open-Source Base

Large companies are not the only players trying to understand where the boundaries are between commercial development and community-based open-source development.

It is of course the great challenge for open source to prove it can create products that are accessible to non-technical users. Many open-source developers are experimenting to find the best way for their work to "cross the chasm" from early adopters to a wider market.

In his contribution to this section, John Ousterhout argues there is a natural complementarity between open source and commercial development. Open source provides raw material, if you will, that can be further refined and extended by commercial development.

Dick Hardt of ActiveState, which creates Perl products for the Win32 platform, makes a similar argument. In a presentation given at a recent Perl conference, Hardt used the metaphor of logs and lumber. Open source, he argued, is a great way to produce logs. But while there are users who are happy to build log cabins, or to split their own logs into lumber, a wider class of users can be reached if companies provide "lumber" in the form of pre-built binaries, value-added interfaces or development environments, documentation, and other elements of commercial products.

Of course, there is a continuing discussion in the open-source community about the best way to provide such commercial development. A company such as Red Hat, which distributes Linux, is firmly committed to the GNU Public License, even for their value-added extensions. Their argument is that packaging, brand, and channels of distribution are sufficient to protect their market and give them an acceptable return on their investment.

Other companies, such as Ousterhout's Scriptics, Allman's Sendmail, Inc., and Hardt's ActiveState Tool Corp., are taking a hybrid approach, providing both a rich open-source product base and proprietary value-added extensions. Strong partisans of the GPL and the strict Open Source Definition (see www.opensource.org/osd.html) might find fault with such efforts. Ultimately, though, experimentation in the market will tell us more than philosophical debates whether an open-source license (or simply an open architecture), access to source code (even if only on the basis of a paid license), or tools for integrating the user community into the development process, are the most important elements in the continuing success and scalability of open-source projects. C

Tim O'Reilly (tim@oreilly.com) is the founder and CEO of O'Reilly & Associates, a leading publisher of books about open-source software. He convened the first "Open Source Summit" in 1998 to bring together the leaders of major open-source communities and has been active in promoting the Open Source Initiative through writing, speaking, and conferences.