A managerial overview of open source software

Sandeep Krishnamurthy
Associate Professor of E-Commerce and Marketing, University of Washington, Bothell, Washington (sandeep@u.washington.edu)

Open source software programs such as Linux and Apache give any interested party access to the source code, leading to a distributed innovation model in which users actively participate in the product's development. Often free, OSS products are distributed under many public licenses, are more reliable, and provide greater flexibility and choice. On the other hand, OSS leads to a proliferation of versions, and may appeal only to high-end users. The system leads to fascinating competitive and cooperative relationships among companies, between a company and a community, and among communities. How can managers choose?

Most commercial software producers consider source code—the raw instructions that determine how a program works—as valuable intellectual property and hence do not make it available to users or the general public. In fact, US law permits software makers to refrain from releasing the source code for copyrighted software for 95 years. However, one category of programs—generally referred to as open source software (OSS)—share their code with any interested user.

OSS programs are now pervasive. There are at least two well-known desktop operating systems: Linux and BSD. Products such as Open Office, Star Office and KDE Office compete with the famous Microsoft Office suite in the productivity software market. Several open source Web browsers are available; the leading examples are Mozilla and Nautilus. The infrastructure of the Internet is run using many OSS products, including: Sendmail, the software used to transport most e-mail on the Net; Apache, the Web server software; and BIND, the program that provides domain name server (DNS) services for the entire Web, translating domain names into IP addresses. Many smaller products are also thriving. As of April 22, 2002, sourceforge.net, an OSS directory, hosted 38,425 programs and had 406,368 users.

OSS must not be confused with public domain software (PDS), which is unconditionally free and not copyrighted. Users of PDS may or may not have access to the program source code. Other forms of free software, such as shareware, browser plug-ins, and media players, may or may not make their code available and may have other restrictions. OSS also differs from protocols (such as TCP/IP) and standards (such as 802.11, an IEEE standard for wireless broadband), which are agreed-upon formats for data transmission that can be granted by institutions or determined after negotiations between firms.

Linux has emerged as the poster child of the OSS approach. According to OpenSource.org (2002b), internal reports at Microsoft revealed that Linux performed better than many MS products on speed and reliability. Micro-

soft has also explicitly labeled Linux a threat to its long-term business. In 1997, Infoworld magazine gave Linux the "Best Operating System" award and hailed its community for excellent technical support. The program is being used in many companies to handle mission-critical steps. A partial list of companies that have adopted it includes names such as Disney, Merrill Lynch, Pixar, the US Postal Service, Siemens, and Mercedes-Benz. In addition, companies such as IBM and Sun Microsystems have announced major investments in OSS products. IBM alone has invested $1 billion toward its Linux-based products. Sproull and Moon (2000) list five factors they believe contributed to Linux's success:

1. *The Internet:* The sheer global scale of the endeavor and the complex coordination involved would not have been possible without the connectivity of the Net.
2. *Modular product:* The modular design made it possible for individuals to contribute by working alone.
3. *Strong leader with clear focus:* Linus Torvalds, inventor of the program, consistently had the support of the community and was able to impose his vision successfully.
4. *Parallel release structure:* Even-numbered releases were stable, while odd-numbered releases were more developmental and had the latest features. This pleased two user groups, which led to greater buy-in.
5. *Consistent support from communities of practice:* The developers who worked on Linux believed in it. Such faith led to an engaged community that provided strong support.

Table 1
Linux's global picture: Number of user groups per country as of July 2003

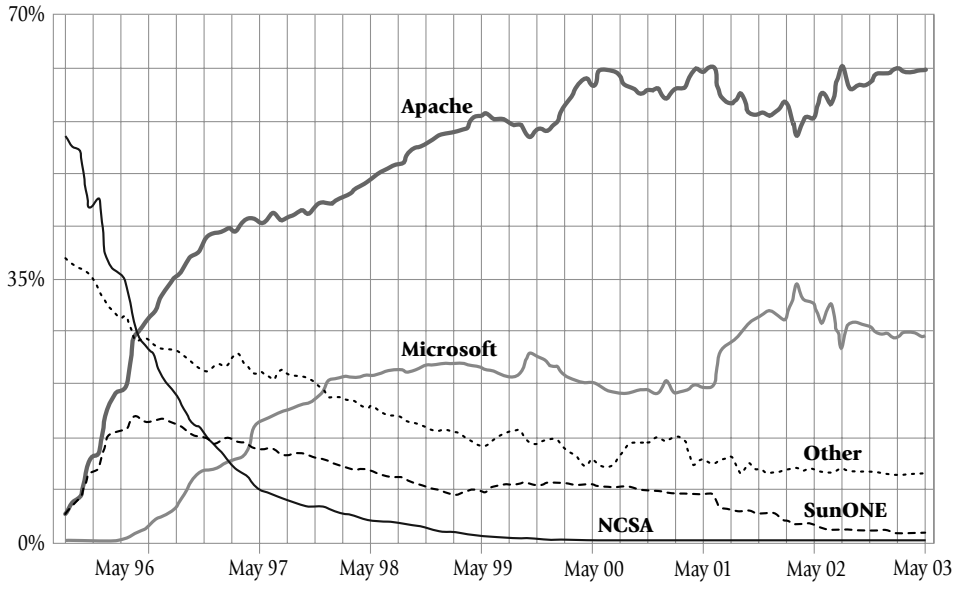
Country	*No. of groups*	*Country*	*No. of groups*
US	143	Australia	9
Germany	46	Belgium	9
India	30	Denmark	8
Canada	25	Mexico	8
UK	23	Norway	8
Italy	22	Switzerland	8
Spain	19	Austria	5
Russian Fed.	17	Colombia	5
France	14	Netherlands	5
Brazil	13	Poland	5
Argentina	11	Sweden	5

Countries w/4 groups	4
Countries w/3 groups	6
Countries w/2 groups	7
Countries w/1 group	36

(Source: www.linuxcounter.org)

OSS products have been quite successful. Consider the Web server application Apache (httpd.apache.org/docs/misc/FAQ.html). A tracking survey conducted by Netcraft shows that Apache is the most widely used Web server software, with a market share of about 70 percent. This is impressive, considering that the main competitor is Microsoft, the largest software maker in the world. A summary of the data of this study is shown in **Figure 1**. (To be fair, Apache is heavily deployed at hosting companies and ISPs that strive to run as many sites as possible on a single computer to save costs, whereas Windows is most popular with end-user and self-hosted sites, where the host-to-computer ratio is much smaller.)

Figure 1
Netcraft's Web server data



(Source: www.netcraft.com/survey)

OSS is a global movement, benefiting developers and users from around the world. The number of Linux User Groups (LUGs) worldwide is a clear testament to this fact—even in countries with very low PC and Internet usage (see **Table 1**). These groups provide informal support to Linux users and act as forums to disseminate the latest information. OSS products have a

strong economic impact. Wheeler (2001) estimated it would have cost more than $1 billion to develop Red Hat's Linux 7.1 version had a proprietary development model been used.

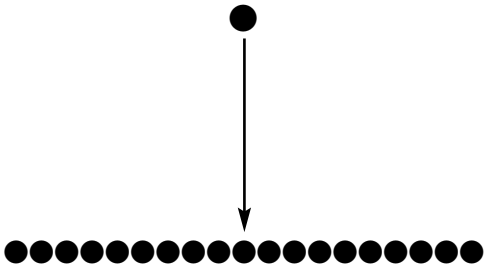
OSS vs. other software types

The key difference between OSS and commercial software is illustrated in **Figure 2**. In the world of software, access to the source code is a prerequisite for innovation. When the code is closely guarded, product innovation is limited to the author. Users may try out the product and provide suggestions for change. But the original author is the sole entity with the ability to make changes to the software. There is a clear demarcation between the roles of the author and the user.

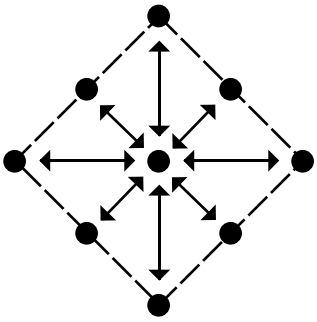
By providing users with access to the source code, on the other hand, OSS empowers them to aid in product improvement. Because anybody can make changes to the product, a large pool of producers is created and the distinction between consumers and developers becomes blurred. Authors ask for help with certain modules, and interested developers provide it. Users can create customized solutions for their needs, adding features or improving on existing ones. When faced with a problem, they can "look under the hood" to try to find a solution rather than relying on external sources. Interested users can even build new products based on the source code.

Figure 2
Key difference between OSS and other software

Commercial software model:
Strict separation between producer and user



OSS development model: Author-user interaction, user-user interaction, larger community



Sun changes the price on StarOffice from $0 to $75.95

A German company called Star Division first developed StarOffice in the 1980s. Then Sun Microsystems acquired the company in 1999. Sun released StarOffice 5.2 in June 2000 and donated the source code to the OpenOffice initiative (www.openoffice.org). StarOffice continued to be available as a free download from Sun's website. However, starting May 21, 2002, Sun decided to start charging $75.95 for the product.

The company views this as a low-cost alternative to Microsoft's Office Suite, whose basic version sells at $499. Sun also announced that StarOffice would be distributed with many Linux versions. However, it is not clear whether the company can compete effectively. Large enterprises that are currently using Microsoft Office will have to retrain employees. Sun's salesforce is used to selling different types of products and there will be no exclusive salesforce pushing this one.

Only time will tell if the move from free to fee was successful for Sun.

(Sources: zdnet.com/2100-1104-913868.html; www.openoffice.org/about.html#history)

Thus, the OSS system is predicated on an interactive relationship with the user. Most corporations have increasingly shielded their developers from market feedback. The logic is that the amount and quality of feedback would interfere with the efficiency of product development. In direct contrast, OSS users have direct access to code authors and can frequently tell them what they do and do not like. Participants suggest product features, test the program, help with the development, and answer the questions of other users. This multiple interactivity is what makes OSS a community-based movement rather than an organizational response to market forces.

Pricing

Most people assume that all OSS products are free. While this is true for most such programs, zero pricing is not necessarily a fait accompli for OSS, as illustrated in the sidebar above. Releasing the source code and pricing are two separate decisions.

It is true that most OSS products are conditionally free. They may be downloaded and distributed at will. There is no concept of piracy for most such products—it is perfectly legal for a user to share the CD of the program with friends. However, restrictions are placed on the use and modification of the source code to ensure the integrity (protect the property rights) of the original creators and to maximize access to the modified product.

Many readers may be unaware that even commercial products could potentially contain code that is taken from an existing open source code base. Some licenses permit this, while others do not. For instance, IBM's WebSphere suite contains code from Apache. (Licensing will be discussed further in the next section.)

From a corporation's standpoint, using already available source code reduces the cost of developing commercial software. There is no need to reinvent the wheel. In fact, several developers judge their success by corporate acceptance of their code.

Microsoft's shared source program clearly illustrates the separation between disclosing code and pricing. Under this program, the company shares its code with enterprise customers, considering it a tool that will help the customers as well as large distributors. However, this is not a free service; the fee for participation is $10,000. Moreover, there are restrictive terms of use. For example, product innovation can still be conducted by Microsoft only, and the code cannot be shared with others.

Choosing a license

Unlike public domain software, open source programs are copyrighted, with the copyright owned by the original authors. And they are covered by a license. *It is important to realize that there is no one standard license.* Rather, there are a variety of licenses, each with its own unique twist. The authors are free to prescribe the set of conditions that users who download the program must adhere to. Indeed, authors who disagree with pre-existing license formats can write their own, though they rarely do. Typically, software authors choose from a set of licenses that are approved by the Open Source Institute (www.opensource.org).

Licenses differ on four key dimensions: (1) whether they allow the OSS program to be mixed with non-free software; (2) whether modifications can be made private and not returned to the author(s); (3) whether the program can be relicensed by anyone; and (4) whether it contains special privileges for the original copyright holder. A comparison of the leading licenses on these four dimensions is presented in **Table 2**.

Perhaps the best known and most widely used among these is the General Purpose License, which covers Linux. The GPL has been criticized for its stringent requirement of asking users to release the source code for any derived work, that is, any work that uses the original code. This stipulation holds only if the derived work is to be sold or redistributed. A derived work of a GPL-covered software program that a firm uses internally need not be made open source. Moreover, if one product is under a GPL, it does not mean that other programs bundled with it automatically fall under the license as well.

The choice of license has a crucial impact on a product's prospects. The restrictive nature of the GPL, for instance, may keep some developers away from the product. To overcome this, a new license called Lesser GPL, or LGPL, retains all the features of the original except the one that stipulates the release of derived works to the public. It is likely that this may attract more developers. As one anonymous programmer put it (Bauer 2002):

> I personally prefer LGPL, so commercial companies can use the code if they help the developers to modularise it enough. Then they are obliged [to] return their changes to the LGPL-ed modules, but their product as [a] whole is…proprietary.…

As mentioned in the discussion on pricing, it is possible to build a program or set of programs around an OSS product and then sell it for a fee—under some conditions. Specifically, it depends on the license chosen by the creators of the product. In certain cases, software authors are content with companies (or others) profiting from their code—provided they get the proper recognition. Some licenses, including the GPL, forbid companies to do this.

The advantages

Several large companies now run open source software. Google, for instance, runs its entire search operation using Linux. What, then, are the advantages of OSS products that make them so appealing to big companies?

Table 2
Comparison of open source licenses

License	*Can be mixed with non-free software*	*Modifications can be made privately and not returned to author*	*Can be relicensed by anyone*	*Contains special privileges for the original copyright holder over user's modifications*
General Public License (GPL)	No	No	No	No
GNU Library General Purpose License	Yes	No	No	No
Berkeley System Distribution	Yes	Yes	No	No
Netscape Public License	Yes	Yes	No	Yes
Mozilla Public License	Yes	Yes	No	No
Public Domain	Yes	Yes	Yes	No

(Source: www.perens.com/Articles/OSD.html)

Free

To begin with, many OSS products are free, though of course the caveats discussed in the pricing section apply. This lowers the risk of trying the product and improves the chances of rapid diffusion among the appropriate target audience.

Given their free nature, OSS products are "viral"—a new product can be passed quickly from one person to the next. Because piracy is not an issue, they are quickly accepted by a global target audience. This viral nature is a key advantage over commercial software.

Large developer and tester base

Traditionally, a company hires a certain number of developers to craft software. Next, a group of testers work with the product to make sure the number of bugs is minimized. Then the program is launched on the market.

With the open source method, however, a potentially endless number of developers and testers can work on the product. In the words of Linus Torvalds (Ghosh 1998):

> There are lots of advantages in a free system, the obvious one being that it allows more developers to work on it, and extend it. However, even more important than that is the fact that in one fell swoop it also gave me a lot of people who used it and thus both tested it for bugs and tested it for usability. The "usability" part comes from the fact that a single person (or even a group of persons sharing some technical goal) doesn't even think of all the uses a large user community would have for a general-purpose system.
>
> So the large user-base has actually been a larger bonus than the developer base, although both are obviously needed to create the system that Linux is today. I simply had no idea what features people would want to have, and if I had continued to do Linux on my own it would have been a much less interesting and complete system.

However, this advantage is not available to all OSS products. Building a community of developers/testers is a challenge for newer products that may get lost in the mix.

Recently, authors such as Weber (2001) and Lerner and Tirole (2000) have suggested that the answer lies in choosing the appropriate area to work on. A creator of a new OSS product must focus on projects that developers would consider important, "sexy," and significant additions to the software universe. Developers are not interested in spending time on products that would lead to a dead end or would have a marginal impact. They focus instead on immediate and tangible problems—what is referred to as the "scratch the itch" strategy.

Peer review equals reliability

Because of its organizational structure, some argue that the open source structure leads to rigorous review, which in turn leads to more reliable products. As hacker and author Eric Raymond explained (Leonard 1998):

> The central problem in software engineering has always been reliability. Our reliability, in general, sucks. In other branches of engineering, what do you do to get high reliability? The answer is massive, independent peer review. You wouldn't trust a scientific journal paper that hadn't been peer reviewed, you wouldn't trust a major civil engineering design that hadn't been independently peer reviewed, and you can't trust software that hasn't been peer reviewed either. But that can't happen unless the source code is open. The four most critical pieces of infrastructure that make the Internet work—Bind, Perl, sendmail, and Apache—every one of these is open source, every one of these is super reliable. The Internet would not function if they weren't super reliable, and they're super reliable precisely because throughout their entire history people have been constantly banging on the code, looking at the source, seeing what breaks and fixing it.

In contrast, when a firm does not make its source code available to the public, it has to employ several testers and developers at considerable cost. Even a company willing to devote resources to product testing may not be able to accurately simulate all the conditions under which it will be used. Therefore, the product does not go through a rigorous enough testing process, which makes it less reliable. In many cases, the company learns from early releases and gradually improves on it. With global user communities, OSS products get tested on a much larger set of conditions, and a much more reliable product is turned out.

The debate about the *security* of OSS products closely dovetails with this point. If there is strong peer review, security flaws will likely be caught. The better the structure of the peer review, the better the security. It may seem that a large number of people with access to code is bad for security. However, because project leaders scrutinize any changes in the code before they are accepted, this turns out to be a good thing. Due to the global user/developer base of products such as Linux, security flaws are caught early and fixes are prepared ahead of corporations. As of this writing, there have been almost no reports of serious security flaws in OSS products, in contrast to the several that are routinely reported in commercial products.

Flexibility of use

One of the problems with regular software programs is that unless you work with all the software from one company, you do not have the flexibility of "mixing and

matching." According to Torvalds (Ghosh 1998):

> In fact, one of the whole ideas with free software is not so much the price thing and not having to pay cash for it, but the fact that with free software you aren't tied to any one commercial vendor. You might use some commercial software on top of Linux, but you aren't forced to do that or even to run the standard Linux kernel at all if you don't want to. You can mix the different software you have to suit yourself.

Stated otherwise, the OSS movement is all about *choice*. For instance, it is possible to run Linux using multiple user interfaces (Gnome, KDE) and browse the Web using different browsers (Nautillus, Mozilla). There are a variety of competitors to Microsoft Office available for Linux, such as Open Office or KOffice.

Customer support from a community

Traditionally, a user with a problem has to contact the technical support division of the software firm. In many cases, the level of support is poor and the firms have consistently reduced the resources devoted to customer service. In direct contrast, anyone using OSS has an engaged community willing to answer questions. All of them have actually tried the product, which makes them more capable of answering questions.

Table 3
Survey of Linux kernel versions

Kernel	*Number of machines*	*% using version*	*Kernel*	*Number of machines*	*% using version*
2.0.28	2	0.1%	2.4.16	39	1.1%
2.0.30	2	0.1%	2.4.17	56	1.5%
2.0.34	4	0.1%	2.4.18	733	20.0%
2.0.34C52_SK	2	0.1%	2.4.19	300	8.2%
2.0.36	5	0.1%	2.4.19acl	3	0.1%
2.0.37	3	0.1%	2.4.19snt	2	0.1%
2.0.38	3	0.1%	2.4.2	40	1.1%
2.0.39	6	0.2%	2.4.20	1,226	33.5%
2.2.10	2	0.1%	2.4.20.1	2	0.1%
2.2.12	10	0.3%	2.4.20usi	3	0.1%
2.2.13	13	0.4%	2.4.21	475	13.0%
2.2.14	34	0.9%	2.4.21.cek	3	0.1%
2.2.15	2	0.1%	2.4.21rel	3	0.1%
2.2.16	62	1.7%	2.4.22	11	0.3%
2.2.16C32_III	2	0.1%	2.4.22_pre2	2	0.1%
2.2.17	23	0.6%	2.4.3	11	0.3%
2.2.18	20	0.5%	2.4.3.cek	2	0.1%
2.2.18pre21	4	0.1%	2.4.4	26	0.7%
2.2.19	103	2.8%	2.4.5	6	0.2%
2.2.19ext3	4	0.1%	2.4.6	4	0.1%
2.2.19pre17	9	0.2%	2.4.7	46	1.3%
2.2.20	65	1.8%	2.4.8	13	0.4%
2.2.20RAID	2	0.1%	2.4.9	42	1.1%
2.2.21	9	0.2%	2.4.x	2	0.1%
2.2.22	17	0.5%	2.5.68	3	0.1%
2.2.23	6	0.2%	2.5.69	3	0.1%
2.2.24	11	0.3%	2.5.70	3	0.1%
2.2.25	40	1.1%	2.5.71	2	0.1%
2.2.5	6	0.2%	2.5.74	6	0.2%
2.2.9	2	0.1%	Others		1.4%
2.4	2	0.1%			
2.4.0	6	0.2%	2.0	31	0.8%
2.4.10	45	1.2%	2.2	453	12.4%
2.4.12	9	0.2%	2.4	3,154	86.1%
2.4.13	4	0.1%	2.5	26	0.7%
2.4.14	9	0.2%	Others		0.0%

Source: Alvestrand (2003). Information from 3,665 machines as of July 12, 2003. Kernels run by only 1 machine are not listed.

The disadvantages

Despite the many positives associated with open source product development, we must also concede a few disadvantages as well. These include a proliferation of versions, an appeal to mainly high-end users, and somewhat limited marketing capability.

Version proliferation

Consider the data in **Table 3**. There is a tendency in open source software to have a proliferation of versions. This is certainly true for Linux. As shown in the table, there are at least 65 versions of the software running on user machines as of this writing. In other cases, there is a phenomenon called "forking the code," which further exacerbates the version proliferation problem by creating two different evolution paths for the product. Managers may have to choose one group over the other, exposing themselves to the potential risk of product failure.

Appeal to high-end user?

One of the criticisms of open source software has been that it appeals mainly to the high-end technical user. That is why products such as Linux have done well on the server side (the "back office") but not so well on the client side (the desktop). Similarly, technical products such as Apache tend to be used by skilled technical workers rather than the lay employee. The mainstream user wants performance and features over reliability, one easy-to-use suite rather than a choice, and probably does not care about access to the source

code. The rebuttal to this criticism has been that OSS products are increasingly becoming user-friendly. Many applications have now been developed to make the user interface to Linux appear very similar to Windows.

Marketing

OSS communities lack the resources to market these programs through traditional media. As a result, the awareness of a new OSS product spreads mainly through word-of-mouth. This makes it easier to build awareness among developers and technical users, but harder to gain mainstream acceptance. Of course, as companies such as Red Hat become more successful, more resources will be available for the marketing of these products.

Businesses and the OSS movement

Enterprises of all stripes are operating today in the OSS space. Here the discussion is limited to the impact of open source on two types of companies: software producers and vendors.

Software producers

Software producers are perhaps the most affected by the phenomenon. They are frequently under attack from free OSS products and are struggling to compete with them. As mentioned earlier, Microsoft has identified Linux as its main competition in the coming years. To compete effectively, these companies have adopted two strategies: *total cost of ownership (TCO)* and *code sharing.*

The idea of focusing on TCO is to look at the firm's total cost rather than the cost of acquiring the product. Added to the shelf price are such cost components as employee retraining, technical support, purchase of additional software, integration with legacy systems, and installation. A software maker must be in a position to drastically reduce the cost of post-sale services—a non-trivial proposition in most cases. This also has interesting pricing implications. Software makers can cut the front-end cost (shelf price) and shift the reduction to back-end expenses (post-sale costs). Thus it becomes important for customers to be able to evaluate the alternatives for themselves when choosing.

In the second strategy of competing with an OSS product, code can be shared among a *private* group of enterprise customers, as with Microsoft's shared source program, or it can be shared *publicly*, with the rest of the world. Public code sharing can be an important strategy when a company is trying to build a critical mass of developers. Lerner and Tirole offer two suggestions for doing this:

- If by releasing the code in one segment of the market, profits will increase in a complementary segment, the firm must make its code open source.
- Releasing the code makes the most sense when the firm is very small or lagging behind the market leader by a significant margin.

OSS vendors/distributors

The revenue model for OSS distributors is based on two sets of revenue streams: delivery cost and post-sale services. First, as shown in **Table 4** on the next page, even though the distribution cost is said to be free, there are subtle interpretations of what "free" actually means. The product is available as a free download from the Net. However, most customers are comfortable with obtaining a software program on a CD, for which distributors charge. As a result, companies can ask as much as $124 for their version of Linux. Of course, some offer even the CD for free. Similarly, OSS vendors sometimes require a payment to make program updates available to customers.

The second set of revenue streams is based on *post-sales service.* OSS vendors operate under the assumption that firms adopting such a product also want accountability. If something goes wrong, they do not want to be kept waiting by members of a global developer community. Rather, they want the problem to be fixed quickly, and they are willing to pay for this service ("one throat to choke"). As a result, vendors offer such services as product installation, integration with existing systems, and customer support—at a price.

As discussed earlier, one problem with OSS products is version proliferation. Most firms do not want to bother with having to pick from the myriad versions available at any point. So vendors play a very important role of *version authentication.* They take over the hassle of going over the different versions and choosing the one they consider the best. The customer simply agrees to buy the version the vendor supplies, with the understanding that the vendor will provide the appropriate customer service. It is clear from Table 4 that not all Linux vendors use the same version of the kernel (the main part of the program). In effect, the different vendors are fighting a standards war trying to impose their desired version on the customer base.

Certification/training/education of developers in OSS products is also an important source of revenue for the vendors. Red Hat, for instance, charges $850 to certify one developer. Publishers such as O'Reilly have done very well selling technical books on OSS products such as Linux.

Vendors frequently offer *premium* or *advanced* versions of their product for a price. This is another source of revenue. For example, Sourceforge.net is a free collaborative space. But Sourceforge Premium is a paid service firms can use to create a collaborative organizational environment.

Table 4
Comparison of major Linux distributions

	Mandrake	*Red Hat*	*Debian*	*Gentoo*	*SuSE*	*Stackware*	*Lycoris*	*Beehive*	*Turbolinux*	*Caldera*
Package	8.2 Download	7.3 Standard	3.0r0 Debian	1.2 Linux	8 Personal	8.1 Linux	Amethyst2 Desktop/LX	0.5.0 Linux	8 Workstation	3.1.1 Workstation
Release date (in 2002)	3/18	5/6	7/19	6/5	4/22	6/182	7/28	4/17	5/31	1/30
Price (US$)	25	60	Free	Free	40	40	20	Free	124	99
Origin	France	USA	Global	USA	Germany	USA	USA	USA	Japan	USA
Support	30-day Web-based installation support	30-day Web-based installation support	Mailing lists	Mailing lists, forums	60-day installation support	Installation support & limited tech support	60-day e-mail support	Mailing list	Unlimited installation & TurboTools support	60-day installation support
Documentation	Installation guide	Installation & RHN ref. guides	Online	Online	Basic & applications manuals	Online	Installation guide	Online	Installation, TurboTools & support guides	Online
CDs	3	7	7	1	3	4	3	1	7	6
FREE *download*	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Linux kernel (2.4.19)	2.4.18	2.4.18	2.2.20	2.4.19	2.4.18	2.4.18	2.4.18	2.4.18	2.4.18	2.4.13
Installation	Graphical	Graphical	Text mode	Text mode	Graphical	Text mode	Graphical	Text mode	Graphical	Graphical
Default desktop	KDE	Gnome	---	---	KDE	KDE	KDE	KDE	KDE	KDE
Office suite	KOffice	KOffice, StarOffice	KOffice	KOffice, OpenOffice	KOffice, StarOffice	KOffice	KOffice	KOffice	KOffice, StarSuite	KOffice, StarOffice

(Source: www.distrowatch.com/top.php)

Other kinds of interaction

Enterprises can interact with the OSS approach in other ways. One is to *facilitate new open source movements.* Remember that in OSS, software programmers are volunteers. They may be paid by their institution in some cases, but in many cases they are not. As a result, there is a dire need for companies that can support new OSS initiatives. For example, companies such as Red Hat provide grants to new efforts.

Firms can also *adopt* open source products. Many large companies are now incorporating them into their existing product lines, such as IBM with the Apache server in its WebSphere suite. By doing so, companies tap into a vast global community of developers and reduce their cost of production by not having to reinvent the wheel.

Many companies have tried to become the number one source of news and information about open source products. Leading Linux sites include CNET, Newsforge, and Screaming Penguin. In addition, firms can encourage the style of open source communities within their organizations. For example, widespread code sharing within a firm is uncommon and may be worth considering.

Competition and cooperation

The OSS arena provides interesting examples of competition and cooperation. Our discussion of this can be divided into three parts: company vs. company, company vs. community, and community vs. community. First, however, we should briefly discuss the nature of an OSS community.

Anybody can develop an open source product; individuals, small groups, non-profit organizations, for-profit companies, and universities have all done so. However, it is most common to find OSS products associated with individuals, small groups, and nonprofit firms. An OSS community is simply a group of developers who share a passion for a product. They get together electronically on mailing lists and discussion groups and use online collaboration tools for working together effectively. The community has no interest in making a profit from the product. Typically, the group is headed by one or more project leaders. In some cases, one might find a "leadership by committee" structure, with different members assigned responsibilities for different portions of the project. This is especially true for large projects. The leader or leaders decide which code gets in and set the vision for the future of the product. Most OSS communities welcome all interested developers to participate in their activities. However, there is a strong implied hierarchy, and it may be months before a newcomer actually contributes to the product's code base.

United Linux takes on Red Hat

On May 30, 2002, four distributions of Linux—Caldera, Conectiva, SuSE, and Turbolinux—came together to announce the formation of United Linux. The goal was to combat the increasing dominance of Red Hat (one survey in 2000 put Red Hat's global share at 70%), especially in the United States.

The coming together of these four companies also points out the problem of fragmentation in open source. Since any company at any time can take the existing source code and simply incorporate it in its code base, several different versions of Linux may exist at any point. By joining together, these companies hope to emerge as a credible alternative to Red Hat Linux.

United Linux has already garnered the support of major corporations such as IBM, HP, NEC, and Fujitsu. However, a lot will depend on how well the companies work together and how focused they are on the end result. The new initiative also raised eyebrows when it said that it would not make the entire source code available for free. Critics have said that this initiative may not work because it has not unified all distributions other than Red Hat; notably, Mandrake has stayed away.

Either way, competition in the Linux marketplace is just heating up.

Source: www.unitedlinux.com

Company vs. company

All companies are in search of a long-lasting competitive advantage. OSS vendors are no exception. There is intense competition among the different vendors.

Consider the example of Linux as discussed in the sidebar above. Red Hat has emerged as the product leader in the US market. Moreover, as Table 4 shows, there has been a proliferation of distributions with a number of vendors trying to sell their version. Customers are at a loss and do not know which version is better. To do well in this situation, a number of small vendors have come together to create one credible alternative to Red Hat.

There is similar competition among vendors in other products. Sun Microsystems is now trying to pit its Star Office against the market leader, Microsoft Office. IBM's WebSphere suite (which includes Apache) is competing with several other rivals.

Company vs. community

All kinds of interesting relationships exist between companies and OSS communities. Microsoft provides good examples of both competition and cooperation. On the one hand, it is engaged in a bitter battle with Linux. Its salespeople have been asked to systematically attack Linux when they make sales calls. It has attacked the GPL as being fundamentally at odds with intellectual property rights. Actively pushing the idea of TCO, it claims a strategic edge over Linux on that attribute. OSS advocates claim that Microsoft has spread a lot of fear, uncertainty, and doubt ("FUD") about OSS products in the marketplace, dissuading potential customers from transacting with a product developed by a community.

At the same time, Microsoft has built a cooperative relationship with Ximian—an open source community. The company is now working with Ximian to build a Linux-based version of its .Net platform. Ximian views this as an opportunity to work on a cutting-edge product and does not have a problem innovating above and beyond what a for-profit company has done. It has a history of working collaboratively with for-profit companies such as Hewlett Packard, IBM, and Corel. Ximian's belief is that this cooperative approach leads to positive results for all.

Community vs. community

OSS products compete with each other all the time. For instance, it is now widely acknowledged that Linux has edged out BSD in the operating system market. The practice of forking is also important here. Many times, a community comes up with two competing visions for the

future. If it is unable to resolve this conflict, it splinters into two groups and the resulting products may become competitors.

The competition among OSS products is important because there is a limited supply of developers. These people will work with winners, and a critical mass of them is needed for success.

The goal here has been to provide managers with an overview that can help them choose between open source software and a commercial software product. OSS products are more reliable, provide greater flexibility and choice to users, and are frequently free. On the other hand, this approach leads to a proliferation of versions and may appeal only to the high-end user. However, the main principles of OSS products—distributed innovation, access to source code, and support communities—are features that everybody can benefit from. Companies must educate their employees about this approach. Purchase managers must expand their set of alternatives to include OSS products. And managers should consider implementing some key principles of OSS within their organizations. ❍

References and selected bibliography

Alvestrand, Harald. 2003. The Linux counter project. @ www.linuxcounter.org (12 July).

Bauer, Kirk. 2002. Conversations: Why I don't use the GPL. *Linux Journal*. @ www.linuxjournal.com/article.php?sid=5935&mode=thread&order=0 (26 March).

Bynari. 2001. www.bynari.com/BCG/cases/mclinux.html and www.bynari.com/BCG/cases/cos.html (September).

Ghosh, Rishabh Aiyer. 1998. FM interview with Linus Torvalds: What motivates free software developers *First Monday* 3/3. @ www.firstmonday.dk/issues/issue3_3/torvalds/index.html

Lakhani, Karim, and Eric Von Hippel. 2000. How open source software works: Free user-to-user assistance. MIT Sloan School of Management Working Paper #4117.

Leonard, Andrew. 1998. Let my software go (interview with Eric Raymond). *Salon*. @ www.salon.com/21st/feature/1998/04/cov_14feature2.html (14 April).

Lerner, Josh, and Jean Tirole. 2000. The simple economics of open source. @ www.people.hbs.edu/jlerner/simple.pdf (29 December).

OpenSource.org. 2002a. Case studies and press coverage @ www.opensource.org/advocacy/case_studies.html.

———. 2002b. Halloween document II @ www.opensource.org/halloween/halloween2.html.

Sproull, Lee, and Jae-Yun Moon. 2000. Essence of distributed work: The case of the Linux kernel. *First Monday* 5/11. @ firstmonday.org/issues/issue5_11/moon/index.html.

Von Hippel, Eric. 2001. Innovation by end-users: Learning from open source software. *Sloan Management Review* 42/4 (Summer): 81-86.

Weber, Stephen. 2001. The political economy of open source software. BRIE Working Paper 140, E-conomy Project Working Paper 15.

Wheeler, David A. 2001. More than a gigabuck: Estimating GNU/Linux's size. @ www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html (30 June; updated 29 July 2002).

The author is grateful to Julie Thornton of Linux International for invigorating conversations about open source. The Open Source Forum run by Karim Lakhani of MIT has been an awesome source of inspiration for this work. Thanks also go to Eric von Hippel and T.R. Madanmohan for their feedback on an earlier version of this article. The author benefited from comments made at a presentation at the Indian Institute of Management and the Indian Institute of Technology, both in Bangalore, India.