# COMMER

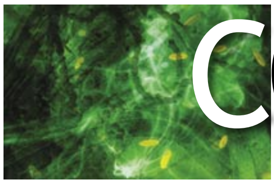
MICHAEL J. KARELS

# CIALIZING open source software

Many have tried, a few are succeeding, but challenges abound.

The use of open source software has become increasingly popular in production environments, as well as in research and software development. One obvious attraction is the low cost of acquisition. Commercial software has a higher initial cost, though it usually has advantages such as support and training. A number of business models designed by users and vendors combine open source and commercial software; they use open source as much as possible, adding commercial software as needed. They may use open source software as a central component of a product or service, but use other components to add value, which can then induce customers to pay for the offering (obviously, it is hard to compete with free software on price).

After a brief overview of the salient differences between open source and commercial software, this article will describe several basic business models in today's marketplace to highlight ways that value is added to open source software and services. For the most part, I will discuss only complete software systems sufficient for some useful purpose, such as network servers, which include an operating system and its associated components, any applications needed for the system's purpose, and necessary local configuration information. Many of the same principles apply to components such as applications and other software packages.



# COMMERCIALIZING open source software

## OPEN SOURCE DEVELOPMENT

The development process for open source software is often quite different from that of traditional commercial software. In some cases a single author or a small group may develop and distribute a program or system. Successful software often attracts additional developers, however, and larger projects generally require larger teams. These teams tend to be distributed, with participants in different locations and with different affiliations. Some members may contribute their own time; others may be paid to work on the project. Some projects develop infrastructure such as a consortium to coordinate the project; others work with a looser organization. In either case, projects are likely to be organized with less central control than in traditional software development. Some projects may have a strong central figure such as the initial author of the software, but many other projects have "outgrown" central control.

This less-centralized structure affects the development process for open source projects in several ways:

- Community support is often available via mailing lists associated with a project. Response ranges from rapid to nonexistent.
- Projects may have many volunteer contributors. Their abilities and availability can vary significantly.
- In terms of quality, Darwinism applies. Some software features may be added while the project is still incomplete or experimental. These features may eventually be removed or replaced, or they may be improved over time. The addition of features and other modifications is driven by the interests and wishes of the contributors (including companies that pay staff to extend open source software). As users of the software, these contributors have certain common interests in making the software stable and usable. They may have substantially different uses for the software, however, as well as different ideas about how the software should be engineered and extended. The direction taken by the software developers may be driven by those who have the most time to devote to development or by those with the greatest tolerance for the discussions on mailing lists for the project. When different groups design and implement the various subsystems, their architectures might not have similar or compatible styles.
- The open source process is inherently social and political. Group leaders spend as much time on organizational matters and conflict resolution as on technical issues. When members cannot agree, the groups sometimes split into different factions. This may result in potentially competitive projects with different approaches, and possibly redundant efforts in both groups.
- The number of features, drivers, and other hardware support tends to grow with time. As the number of users increases, so does the number of developers. Successful open source projects tend to have a long list of features, often including alternative implementations of some components. Hardware vendors provide drivers for the most popular operating systems. These drivers may be well-supported by the vendor, but sometimes they do not enable the most proprietary features of the hardware or may contain obfuscated code for those features to avoid revealing information about them that might reduce a competitive advantage.

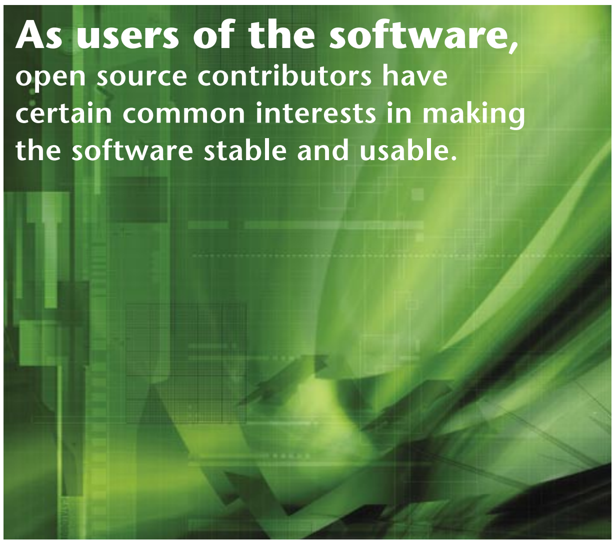
## OPEN SOURCE LICENSING

Licensing is one obvious area of difference between open source and commercial software developers. Navigating that landscape, with its multitude of licenses—each with its legislative intricacies—can be tricky. The following is a brief summary of licenses:

- A small amount of software is in the public domain, in which case there are no restrictions on its use.
- More common is the GNU General Public License (GPL). The GPL allows distribution and modification, but requires that the source code be available without restrictions, along with any binary versions that are released or sold. In particular, modifications to source files covered by the GPL and other code linked to the same program are generally covered by the free source availability terms of the GPL. (See my discussion of the commercial enhancement of open source software later in this article for more on combining GPL and non-GPL code.)
- The BSD-style license, named after Berkeley Software Distribution, uses a copyright owned by the author of

the code or an organization and a notice allowing modification and redistribution in source or binary form with a few limitations, including disclaimer of warranty.
- It is difficult to sell commercial derivatives of programs initially released under the GPL, as well as incorporation of GPL code into commercial applications. Libraries distributed under the GNU Lesser General Public License (LGPL), do not have this property of the license. The GPL does not restrict modifications for internal use, only those provided to others.

Open source software is free to download and has no licensing charge for installation on multiple systems or redistribution.



**As users of the software,** open source contributors have certain common interests in making the software stable and usable.

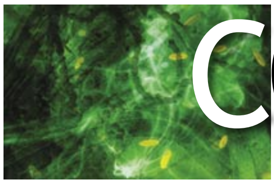
## COMMERCIAL SOFTWARE DEVELOPMENT

Whereas open source software is often developed in a distributed environment, commercial software development is usually under centralized control within a company. This makes it easier to develop a roadmap for the product, control the architecture in a design phase, and possibly maintain an achievable schedule. Commercial groups experience greater pressure to provide features requested by their customers and to set their priorities accordingly. Because commercial developers control staffing instead of relying on volunteers, they can match projects and schedules to the available staff. This makes it easier to handle large or pervasive changes such as operating system support for multiprocessing. More central control also provides some level of accountability to the customers for the design, quality, and usability of the software, as well as legal accountability for the intellectual property rights to the software.

Other relevant properties of commercial software include the following:
- Support is normally available for commercial software, along with services such as training and consulting.
- Because fielding support requests adds expense, commercial developers have an interest in providing documentation, testing user interfaces, making installation and upgrades easy, and otherwise attempting to reduce the need for support.
- The size of the engineering staff at most software companies is finite and typically is never large enough for all of the demands. The number of developers for most commercial projects is smaller than the number of volunteers available for the many open source projects, although the volunteers are generally not full-time. Furthermore, operating system developers face continuous change in computer hardware, as well as network and application protocols that require substantial resources just to keep up.
- Commercial software is most often delivered in binary form, which simplifies installation. Source code is also available for some commercial systems, however. This is especially desirable when customers want to customize or modify the code (e.g., in embedded or specialized systems). Source code is also useful as the most detailed documentation available.
- Commercial software, for the most part, is provided in a package with installable media and documentation and usually requires a licensing fee for each system on which it is installed. Support and updates may require additional fees.

## INSTALLATION AND INTEGRATION

The installation and use of software usually requires some integration work. The developer may do this work, or some or all may be left to the user. Integration includes:
- **Selecting software.** This includes specific packages or programs and the appropriate version for the task. Open source software groups often allow download of the version still in development, the most recent release, and perhaps the most recent stable release if the current release is considered experimental. Although a stable version might be desired, some users may require features found only in newer versions.
- **Obtaining software.** This is either by download or on physical media.
- **Compiling software.** If the software is provided only in source form, binaries must be built. The process can include selecting configuration options for the target



operating system, hardware architecture, alternative implementations for areas such as file locking, and resource limits or other local choices. Some options must be set correctly for the application to build correctly. Others may be necessary for correct or optimal operation. Some packages may depend on other packages—such as libraries, compilers, or interpreters—or cooperating components, in which case a similar process is needed for the required components. Operating systems may require a configuration specific to the target hardware.

- **Configuring software.** Once the software is ready to run, it may require local runtime configuration information. This might consist largely of defaults, have some options specific to the target platform, and require local customization such as directory names, host names or addresses, per-user configuration, etc.
- **Installing and testing.** If the software fails to operate correctly, it must be debugged. The process may be as simple as noting and interpreting error messages. It might also involve analysis of crashes with a debugger, generation of system calls or other traces, and generation of patches as required. Many problems can be traced to incorrect configuration for the target system. Problems installing operating systems are often a result of misconfigured or broken hardware, which can be difficult to debug.

The integration process repeats when it is necessary or desirable to install a new version of the software, install patches, or switch to a replacement program. During the update process, the configuration must be revisited, as options often change. Some packages may provide update scripts that help to import previous configuration information.

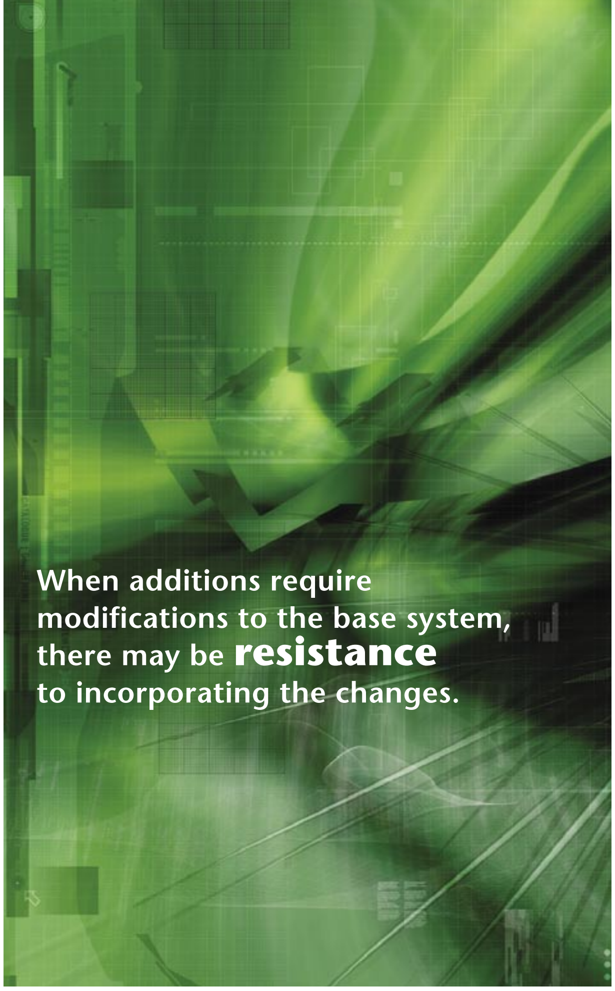
Commercial software packages usually provide prebuilt binary programs and installation procedures, eliminating some of the previously described steps. Open source software may also provide prebuilt binaries, especially operating systems with associated file systems and programs that are difficult to build if a previous version is not installed. Open source packages are more likely to require rebuilding components to select customized configurations or options.

## COMMERCIAL MODELS FOR OPEN SOURCE SOFTWARE

These features of open source and traditional commercial software illustrate some of the differences between the two, as well as uncovering a potential gap between them. Corporate users are accustomed to paying for software and may be unwilling to forgo some of the features of traditional software. This gap has led to a number of business strategies in which companies add features or services to open source software, which they can then sell. The goal, of course, is to take advantage of the resources available through open source and then add value that can generate revenue. In theory, this could provide equivalent value to traditional commercial software at lower cost to the end user. These business models include:

**Distribution.** Walnut Creek CD-ROM, one of the earliest businesses to leverage free software, compiled public domain and freely available software for sale. Although the software was available for download, many users did not yet have access to the Internet or had slow dial-up connections. The value in this model was the compilation of packages and easier access. This business model was compromised by wide access to the Internet, especially with higher-speed access.

**Software integration.** This model changed somewhat with the arrival of open source operating systems such as Linux and FreeBSD. Early distributions contained not only source code for the operating system and included programs, but also ready-to-install binaries needed to get the system running. These releases were gradually enhanced to include more convenient packaging and installation software, which was a significant improvement. Additional open source packages were generally integrated, including the configuration, building, and installation of binaries. A number of Linux distributions, such as Red Hat, competed in this arena. These distributions were generally inexpensive, but did not include much support. Like earlier distribution of free software, the operating system software was free and available for download, but was larger and harder to install. As download speeds became faster, the downloads again competed with CD-ROM sales. Some open source groups, including


When additions require modifications to the base system, there may be **resistance** to incorporating the changes.

FreeBSD and Linux, now even provide easy methods to install while downloading, reducing the need to purchase a CD-ROM; CD-ROMs, however, are convenient, provide a ready archive and backup, and can be used without network access.

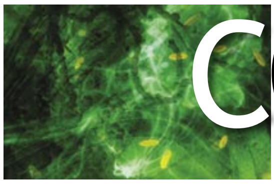
**Hardware integration.** A few vendors, such as VA Linux, also began to provide complete integration, with software pre-installed on hardware selected for compatibility and suitability to the operating system. Larger PC vendors eventually followed suit. IBM has entered the market in a big way, offering Linux on its entire line of hardware. Last year it reported making $1.5 billion selling hardware running Linux (see http://www.cioupdate.com/news/article.php/1574431).

**Support.** Although most early distributors did not include support with their packages, adding support service was the next obvious step. Distributors and consultants began to offer support services using different pricing, including bundled support for a limited time. Support might be limited to installation and stability problems or might include help with configuration, customization, or other services. One issue with this model is that the support organization doesn't control the software that it supports. A possible answer to a support request is thus, "Sorry, that's just the way it is; we can't fix it." The companies providing support, however, often include developers who can generate fixes and will provide new code to the customer—also committing the fixes to future releases of the open source project. The feedback from the support channel is still less likely to influence the software project in major ways, however, because the primary developers don't bear the cost of support and other developers may not approve a short-term fix generated for an immediate problem.

**Publications.** Another highly successful area in the open source market has been the sale of documentation and books. O'Reilly Associates began to print existing open source manuals and in the mid-1980s began a successful series of original books about open source packages. It also expanded into training services via conferences and tutorials.

**Contract development.** This is an obvious extension of the support model, in which a company offers engineering services under contract. Examples range from ports of compilers and operating systems to new CPUs and systems, to writing device drivers and adding custom features. Cygnus Solutions, later acquired by Red Hat, began as a support group for GNU software and added contract work. More recently, Wasabi Systems began using this model. Customers might pay for these projects solely for their own benefit, or they might allow the resulting code to be contributed back to the open source project. (If the original code is subject to GPL and the software is to be sold, the source must be made freely available.) Donating the code improves the likelihood that it will be maintained through future system changes, which reduces the work needed to reintegrate additions into future versions of the base system. When additions require modifications to the base system, there may be resistance to incorporating the changes, but companies with close ties to the open source group are more likely to have their changes accepted. Hardware vendors may use contract development to provide support for their hardware in open source systems. Customers can use contract development to fill gaps in existing software without being dependent on the development process of the open source group, which may have volunteers and unpredictable schedules.

**Commercial value-added.** The previously mentioned models generally add services and integrate additional open source software with the existing free base, in which case the product remains open source. Another option is to bundle commercially licensed software with the open source base, which remains free (e.g., bundling commercial applications with a free operating system). Red Hat added products of this type, which sell at a



# COMMERCIALIZING open source software

## A Foot in the Door: Can Open Source Find Traction in Government?

JOHN M. WEATHERSBY, JR.

The U.S. government is the largest purchaser of information technology products and services in the world. In 2003, the government will spend approximately $60 billion buying, updating, managing, and maintaining an expansive network of computer products and programs.[1]

Open source software seems a likely choice for budget-strapped bureaucrats; indeed, the current climate seems ideal for open source to grab a bigger piece of the government's IT pie. Although open source software is no longer considered an outsider inside Washington, it continues to face challenges on the road to official acceptance as a major player in the government's collective IT system.

### WHAT'S OLD IS NEW

Open source has a long (but quiet, and sometimes unofficial) history as an integral part of civilian and Department of Defense (DoD) networks. Programs such as Linux, Berkeley Software Distribution (BSD), Apache, Samba, MySQL, PostgreSQL, Perl, Python, and Zope are common in most government IT systems. As is often the case in corporate America, most of the support for open source comes from the working ranks of programmers and administrators. Users reported early success with open source programs within the Department of State, Department of Commerce, General Service Administration, and the Postal Service—not to mention numerous programs at National Aeronautics and Space Administration (NASA), the U.S. Naval Oceanographic Office (NAVOCEANO), and, of course, the longtime support of the government's research community.[2] This first attracted attention from the managerial staff and then curious inquiry from policy makers. The concept of open source ("You mean we don't have to pay for this?") continues, however, to present a quandary for many in policy-level positions.

Public-policy makers generally perceive open source software as a dichotomy. On the one hand, open source offers a unique opportunity: Free software and full control of the development and management of IT systems because the source code is included. This can result in increased technological efficiencies, as well as significant financial savings. In addition, open source helps diminish the reliance on any one vendor for service or support.

On the other hand, policy makers see open source software as a disruptive technology because it disregards established development regimes and fails to provide an extensive vendor network that can be held accountable for crashes, updates, and quality control. They want to know exactly whom to yell at when things go wrong (someone who works for a company that the policy folk have heard of). This isn't limited to government—corporate adoption of open source has faced the same hurdle.

### FORTUNE 500 TO THE RESCUE?

The recent flood of support and product offerings by major IT vendors, including Hewlett-Packard, IBM, Intel, and Oracle, has tempered the argument over product support. This increased commitment from large corporate vendors has, as you might expect, significantly raised the interest level of many government officials.

In addition, increased technical recognition provided by numerous government-sponsored reports has helped stimulate interest in open source software and has encouraged agencies to explore and even adopt open source policies.[3]

### MOUNTING MOMENTUM, CONTINUING CHALLENGES

Recent internal studies performed by the MITRE Corporation for the DoD identified more than 100 open-source software applications being used within the DoD.[4] Studies performed for members of the intelligence community and military agencies have highlighted a variety of open source software used within their IT systems, as well as documenting benefits derived from the software.[5]

The official adoption of open source programs throughout the DoD has been limited, however. One reason is higher price than the basic releases of Linux.

A second example is to provide extensions or a professional version under a commercial license while maintaining support for the original free version. Sendmail Inc. is but one example of this. A company using this model would generally contribute to the development of the free version to generate good will, improve the acceptance of the base version, and increase the market for its commercial offering. The extensions might include configuration and management tools, higher-performance or

that, according to a recent federal mandate,[6] all software programs interacting with any system related to national security are required to achieve stringent DoD certifications. Key open source projects are engaged in the various DoD certification processes,[7] but the lack of existing fully certified open source software has hampered adoption efforts into critical defense systems thus far.

The DoD is optimistic that the certification will be attained and that open source has a future within its systems. DoD chief information officer John Stenbit recently issued a policy statement, "Open Source Software in the Department of Defense," which acknowledges the DoD's "current policy [on open source software] and provides additional guidance on the acquisition, use, and development [of open source software] within DoD."[8]

The release of this statement was heralded by many in the open source community as a sign of a leveling of the playing field in the contest between open source and proprietary programs within DoD.[9] It is also assumed that this statement provides a permissive nod to those inside DoD who wish to explore and/or implement open-source programs—while continuing to require open source software to meet the same standards and required certifications as commercial software.[10]

Things are looking up.

## REFERENCES

1 "The Feds Love Linux," Erica Brown, Forbes, June 20, 2003, http://www.forbes.com/2003/06/20/cz_eb_0620linux_print.html.

2 "OSSI Works with Navy," OSSI, http://www.oss-institute.org/ossinavy.html, (for password, e-mail questions@oss-institute.org).
"Open Source Permeates Navoceano Systems," John Lever and John Weathersby, *CHIPS*, Summer 2002, http://www.oss-institute.org/newspdf/CHIPSarticle.pdf.

3 "A Business Case Study of Open Source Software," Carolyn A. Kenwood, MITRE Corporation, July 2001, http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/kenwood_software.pdf.
"Open Source Permeates Navoceano Systems," John Lever and John Weathersby, *CHIPS*, Summer 2002, http://www.oss-institute.org/newspdf/CHIPSarticle.pdf.
"Open Source Software (OSS) in the Department of Defense (DoD)," John P. Stenbit, Department of Defense, May 28, 2003 (PDF copy of the memo), http://www.egovos.org/pdf/OSSinDoD.pdf.

4 "Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense," MITRE Corporation, January 2, 2003, http://www.egovos.org/pdf/dodfoss.pdf.

5 "A Business Case Study of Open Source Software," Carolyn A. Kenwood, MITRE Corporation, July 2001, http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/kenwood_software.pdf.

6 National Information Assurance Acquisition Policy: "National Security Telecommunications and Information Security Systems Committee Fact Sheet No. 11," January 2000, http://niap.nist.gov/niap/library/nstissp_11.pdf.

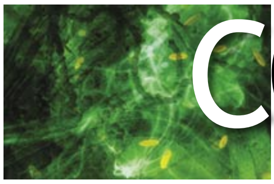
7 "OpenSSL Enters Evaluation for FIPS (Federal Information Processing Standards) 140-2 Certification," Open Source Software Institute Press Release, April 28, 2003, http://www.oss-institute.org/newspdf/OSSI-OpenSSL-fips_PR.pdf.
OpenSSL FIPS Cryptographic Module by Open Source Software on website, April 28, 2003, http://www.oss-institute.org/newspdf/OSSIFIPSRef.pdf.

8 "Open Source Software (OSS) in the Department of Defense (DoD)," John P. Stenbit, Department of Defense, May 28, 2003 (PDF copy of the memo), http://www.egovos.org/pdf/OSSinDoD.pdf.

9 "Stenbit Tells Open Source Users: Check That Legality," Joab Jackson, Washington Technology, June 3, 2003, http://www.washingtontechnology.com/news/1_1/daily_news/20857-1.html.

10 National Information Assurance Acquisition Policy: "National Security Telecommunications and Information Security Systems Committee Fact Sheet No. 11," January 2000, http://niap.nist.gov/niap/library/nstissp_11.pdf.

**JOHN M. WEATHERSBY, JR. is the founder of the Open Source Software Institute (http://www.oss-institute.org). The nonprofit organization's mission is to promote the development and implementation of open source software within federal and state government agencies and academic entities.**



higher-capacity versions, or components that add features such as virus checking in e-mail.

A third example of this model is the embedded systems market, in which a vendor provides development tools for use with a free embedded operating system.

In all three cases, the product now contains commercially licensed and open source software. Customers license and pay for the commercial versions as a traditional software product. One challenge in this business model is to maintain differentiation between the free and commercial versions. The free version must be sufficiently functional to gain market share. The commercial version must have sufficient additional value to induce customers to buy it and to reduce the likelihood of a free knockoff of the added components. Companies that use the commercial model also provide support services, documentation, and training—as would a traditional commercial software supplier.

**Dual license (poison pill).** A clever variation of the commercial value-added model is available for a small class of software, specifically packages that are linked with applications. The Berkeley DB back-end database library from Sleepy Cat Software is one example. In this model, a commercial developer can make a package available for noncommercial use under a license such as the GPL, also providing the same package with commercial licensing and services.

This "twist" is made possible by the requirement to make unrestricted source code available for any application linked with the freely available version—the "poison pill." Companies building products using the back-end database library are unlikely to offer source code, and thus are required to pay for the package. Meanwhile, free software, research, prototyping, and even internal use may be permitted with a free downloadable version of the library. This model is available if the developer owns the original work or if the developer enhances free software that is not covered by the GPL.

**Commercial enhancement of open source.** This model is similar to the commercial value-added model in that a base of open source software is extended commercially, but in this case the base software is modified rather than kept separate. The BSD/OS operating system is one example. BSDi developed the product using the Net/2 and 4.4BSD-Lite releases from Berkeley. It then filled in missing components and made many other modifications that were licensed; added other commercial products; and provided a complete, supported release sold with a standard commercial license. The system is thus a derivative work rather than a bundle of open source plus commercial software.

Of course, such a use of modified code is subject to the licensing of the starting system. In particular, the GPL enforces unrestricted source availability for modifications that are to be included in products. Thus, this type of commercial derivative is not permitted. The line between modifications for which source must be provided and which additions are sufficiently separate from the original GPL software is rather fuzzy, and some companies seem to skirt the limits of the GPL. For example, some vendors provide non-GPL commercial products that are dynamically linked with the Linux kernel (which is covered by the GPL) or are shipped as binary modules for their customers to link into Linux. [For an example of this take on the GPL, read Jim Barton's "From Server Room to Living Room" on page 20 of this issue.]

A product of this type might be derived from a specific version of an open source system, which then diverges from the open source; or it might track the open source version, merging with selected versions. Tracking an open source version allows new features to be incorporated. A company could choose the versions of the open source version to be supported (e.g., choosing and testing the versions most likely to be stable for the target uses). On the other hand, merges can be difficult because of the extent of changes and/or incompatible choices. If the open source project continues development and the commercial version doesn't track it, a competition may be set up between the two versions. A company may have difficulty competing with open source in quantity of features, including hardware support, but may compete with support, stability, or specific features. As with the previous model, the commercial version must compete with the free version, and must offer advantages such as performance, stability, support, and the general characteristics of a commercial product with central control.

**Specialized products.** One class of products is a special category of the previously discussed commercial enhancements of open source software. Specialized servers or Internet appliances such as firewalls or load-balancing servers are often built from standard systems, plus the specific modifications for the application. The base operating system could be an open source system or a general-purpose commercial/open source hybrid. Some vendors choose open source systems because they intend to modify the system and thus don't expect support; others choose commercial systems because they want hardware and other support to be provided. This category is worth mentioning separately because it often exists in a hardware/software product with specialized functionality, such as a traditional embedded product with a more advanced system. This category tends to sell for the highest price, as people seem more willing to pay for customized single-purpose appliances.

If the open source project continues development and the commercial version doesn't track it, **a competition may result.**

## NO FREE SOFTWARE

Looking at the software development process, integration issues, and numerous business models for open source software and commercial hybridization, one observation becomes clear: *There is no such thing as free software.* Sure, you're free to download the software and you're free to modify it, but it's important to remember that both volunteers and employees have contributed their time to make available no-cost software for other users. Some projects, including the original BSD releases, were funded by grants from government or industry to develop software that was made widely available. In addition, integration of free software components involves a cost that must be considered. Many organizations and individuals fail to recognize the significant time and cost integration work can take. And, of course, initial cost and integration are not the end of the story: Updates, patches, version upgrades, and technical support are an ongoing overhead.

During the dot-com boom, companies rushed to make a business out of open source software using some of the models just discussed. Some of these companies have fallen by the wayside and others have changed their business models. Making a business of open source software can be difficult. Although the use of freely available software can accelerate the development of a product, it also lowers the barrier to entry for potential competitors. Developers of open source projects may choose to implement features found in commercial products, thereby increasing their attraction. As some users are highly sensitive to price, or prefer to use open source software on principle, they might even select an incomplete open source solution, improving the software themselves or waiting for future versions to evolve. Many companies, however, are willing to pay for software and services that help solve their problems. They take advantage of open source software, and at the same time don't mind paying for integration, support, and services, such as the addition of required features.

Only time will tell which business models ultimately will be successful. Q

**LOVE IT, HATE IT? LET US KNOW:**
queue-ed@acm.org or www.acmqueue.org/forums

**MICHAEL J. KARELS is a consultant and author specializing in BSD and Unix systems. Previously he was responsible for BSD/OS as system architect and VP of engineering at BSDI, and then as principal technologist at Wind River Systems. For eight years Karels was the principal programmer of the Computer Systems Research Group at the University of California at Berkeley, where he was the system architect for the 4.3BSD and 4.4BSD releases of Berkeley Unix. He is a co-author of the book *The Design and Implementation of the 4.4BSD Operating System* (Addison Wesley, 1996).**