Open Source Software: Risks and Rewards

v2.0, 03 August 2005

Author(s) Gary Hein

Conclusion

Open source software (OSS) brings new opportunities and risks to information technology (IT) vendors and customers. Software commoditization is being profoundly affected from the infrastructure layer and moving up to the application layer. As a result, OSS is changing the strategies and business models of hardware and software vendors and system integrators. Customers should understand the risks and rewards of OSS, and should formulate strategies that bring OSS benefits to their IT departments while mitigating the business and legal risks.

Synopsis

Open source software (OSS) is more than just free software. It's a development process, a distribution model, and a set of new software licenses. And, as characterized by OSS proponents, it's a movement. OSS projects, such as Linux and Apache HTTP server, are commoditizing enterprise and Internet infrastructure, and thus pose threats for some vendors while creating new market opportunities for others.

OSS brings more to information technology (IT) than just free software. It grants freedom from vendor lock-in and application churn, freedom to inspect, modify, and improve source code, and the freedom to influence or contribute to projects key to a company's survival. But these freedoms come at a price. Although OSS is free, it isn't always less expensive to implement due to migration, support, training, and maintenance costs. OSS projects add new twists to business risks such as vendor viability and stability, and legal issues remain uncertain as many popular licenses have yet to be tested in a court of law. The risks associated with unproven open source licenses and a lack of indemnification (protection from external intellectual property claims) cause many organizations to shy away from using OSS.

As OSS projects mature, some become "good enough" for a particular task or audience, yet deliver a 10x improvement over the market incumbents. The 10x improvement is often price (either the software or underlying hardware), but it may be perceived freedom from proprietary vendor lock-in, a faster time to market, or the benefits of a developer community.

Commercial software vendors are starting to feel this effect and are responding with pricing and packaging alternatives to make open source alternatives less appealing. At

the same time, OSS vendors are searching for viable business models. The line between commercial software and OSS is blurring, with commercial software vendors now offering "free" alternatives and access to source code, while OSS vendors are implementing pricing models that, in some cases, are more expensive than non-free alternatives.

Analysis

Burton Group originally published the *Application Platform Strategies* overview, "Open Source Software: Risks and Rewards," in October 2003. The initial overview covered many of the basics, including the core licenses, open source business models, benefits and risks, and implications for proprietary software vendors. It also described how open source software (OSS) was a key factor in software commoditization, starting with infrastructure services and moving up the software stack toward user applications.

Now, almost two years later, it's time to revisit OSS, the licensees, and the impact on the vendors, and to update recommendations for enterprise clients. OSS has matured, with stronger integration and support offerings than those available just two years ago. As originally predicted, it has had considerable impact on the strategies of many vendors, including Sun Microsystems, Microsoft, IBM, and Novell. New companies have emerged to address intellectual property concerns or to provide better integration among projects. Open source has placed considerable pricing pressure on market incumbents, often reducing prices or stimulating proprietary vendors to be creative in their product packaging and/or pricing. It's also changing the very essences of traditional software companies, forcing a move from high-margin software products to lower-margin service businesses.

OSS and proprietary software are no longer at opposites ends of the spectrum; many OSS products have adopted proprietary software pricing for updates and support, while proprietary software products and vendors have embraced OSS philosophies and have reduced prices or removed other barriers to proprietary software solutions. There's less of a chasm between the two worlds today than there was just two years ago, and the line between OSS and proprietary products continues to blur.

OSS is not a stand-alone product category. It affects many aspects of enterprise architecture and information technology (IT). As such, Burton Group provides core coverage in this overview that can be applied to almost every open source project, but also examines implications throughout our coverage where applicable. For example, several Burton Group *Application Platform Strategies* documents have examined OSS alongside of proprietary offerings, while other documents have been strictly focused on OSS products or product categories dominated by OSS solutions. In other words, OSS shouldn't be considered a separate product category, but rather a software license and business model that permeates many product categories and architecture components.

What Is OSS?

In the simplest definition, OSS gives everyone full access to source code. Anyone is free to download, inspect, fix, modify, extend, compile, create, and use the software without charge. This simple definition shows why OSS is sometimes referred to as "free software"—there aren't any requirements to pay the software license fees or royalties

that are typically charged for proprietary software products.

But there's more to OSS than just source code access. OSS also describes a development process, a distribution model, several software licenses, and, as characterized by OSS proponents, a movement. Collectively, OSS is a disruptor that is changing the business for hardware and software vendors as it rapidly commoditizes the infrastructure of IT.

A discussion of OSS generally involves references to Linux, so it's important to clarify the relationship between Linux and OSS. Linux refers to the Linux kernel, but is commonly used to describe an entire Linux distribution (such as Red Hat or SUSE). These distributions combine the Linux kernel with hundreds of other OSS packages, including the Apache HTTP server, GNU utilities and toolchains, Samba, JBoss, Tomcat, and OpenOffice. Thus, when vendors, businesses, and customers talk about Linux, they're often referring to the complete Linux distribution, not just the kernel. Likewise, OSS isn't just Linux; there are over 100,000 open source projects in various states of development.

OSS Is More than Just Source Code

Conversations regarding OSS often involve the words "free" and "freedom." It's pretty clear that OSS is free from an acquisition standpoint; if the source code is open and available, anyone can build binaries without paying royalties—that much is obvious. But free also extends to other, less obvious yet equally powerful benefits. Note that "free" and "open source" are often used interchangeably, but that's not always accurate, especially from the perspective of the Free Software Foundation (FSF). For example, software may be released under an open source license, but that license may not ensure that the software (and especially enhancements) is truly free. Case in point: Apache projects are considered open source, but because the Apache license does not require modifications to be given back to the community, the FSF does not consider this software "free."

Most enterprises that use OSS will never modify or build their own internal versions of the open source projects, but rather implement existing binaries that are packaged as part of a larger distribution (typically one of the enterprise Linux distributions). Thus, some may be quick to dismiss the benefits of "freedom to modify" if their enterprise is unlikely to make those modifications. Yet, because OSS source is readily available, it's much easier for *someone else* to experiment, improve, modify, enhance, or combine with other work. And because many OSS licenses require those modifications to be released back to the community, the entire community benefits.

OSS can free users from vendor initiatives and software churn. It's not uncommon for proprietary vendors to stratify their products by artificially limiting performance or hardware support. OSS, by its very nature, is transparent; not only would it be difficult to implement these limitations in OSS, there's no incentive for the OSS community to take these measures. OSS also frees consumers from vendor initiatives that may benefit the vendor more than the consumer. For example, a vendor may determine that a new, proprietary file format will lock in customer data and make switching to a competing product more difficult, even though the new file format doesn't solve a customer

problem.

OSS doesn't tie a developer to a particular vendor's technology or solution. A developer is free to pull from multiple projects and build best-of-breed solutions. Developers like that the transparency of OSS allows a problem in a library or function call to be examined and debugged. Likewise, for many developers, access to source code provides the best documentation possible and the ability to self-service problems and enhancements to software. If a problem or enhancement opportunity is found in the underlying OSS source code, developers are free to modify the code as needed, with the knowledge that if their contribution is necessary and worthwhile, it is likely to be carried forward with future releases.

OSS may provide additional security benefits, but one can't judge the security of a project based solely on the license. It's been argued many times that OSS projects are more secure than closed-source projects, because, in theory, multiple eyeballs can inspect the source code. Access to source code allows checking for intentional security issues such as back doors and unintentional security problems like buffer overflows or weak encryption. While the transparency of OSS would appear to make it more secure, there's little hard data to substantiate that claim. Burton Group believes that OSS development is as sound as closed-source methods, so enterprises can confidently use these packages for important business functions. For additional details, see the *Security and Risk Management Strategies* overview, "Securing Open Source Infrastructure."

OSS Freedom Carries a Price

OSS offers many advantages over more traditional proprietary software products. However, like all tradeoffs in life, these advantages bring costs and risks that must be considered before embracing OSS.

Cost is often cited as the primary motivator for using OSS. After all, if the software is free, doesn't that mean that IT saves money? Not necessarily. Software acquisition costs are usually a small part of any IT solution; in many cases, software costs are typically less than 5% of the total cost of ownership of a particular project. Jonathan Schwartz, Sun's President, has often equated OSS or free software as "…free as in free puppy." In other words, although the puppy is free, food and veterinarian bills (and the occasional damaged household article) are not—just ask any dog owner for validation of this theory.

The same is true for OSS projects. Expenditures for design, deployment, day-to-day management, and maintenance account for the lion's share of software deployments regardless of the underlying software license. And integration costs are typically higher, as implementations that span multiple open source projects typically come with the "some assembly required" label.

Yet price has played a considerable role in the adoption of open source products, most notably Linux, but from a hardware cost perspective, and only when other requirements were met. Almost every customer interviewed for the *Application Platform Strategies* report, "The Enterprise Linux Ecosystem: Free Software, Cheap Hardware, and Expensive Support," cited inexpensive Intel-based hardware as the primary motivator for implementing Linux as a platform for other open source products. Open source

packages, such as Apache, JBoss, or Tomcat, provided “good enough” services running on Linux on Intel servers.

Support remains a challenge for enterprises switching from proprietary software products to OSS. Unlike proprietary software products, there may not be a single vendor to call when things go wrong. But this is largely a cultural and behavioral issue found when switching from proprietary products to OSS solutions. There are abundant OSS support resources on the Internet in the form of Internet mail lists and archives, discussion forums, and support repositories or databases. In fact, at times, it can be overwhelming. A simple question may result in multiple conflicting answers with no authoritative source.

Many OSS users interviewed by Burton Group have been pleased with OSS support resources. For example, several customers interviewed for the *Application Platform Strategies* report, “The P-Languages: PHP, Perl, and Python for Enterprise Scripting,” were satisfied with the level of support they received from the open source community mailing lists. In fact, several of the companies and developers interviewed said that the responsiveness and accuracy of support found on mailing lists was far superior to what is normally associated with commercial vendor support. For example, a major power utility, which uses Perl for 85% of its infrastructure, said that no organization except Cisco Systems provides support that is as responsive and helpful as the open source community mailing lists.

For most OSS projects, the best place to look for support is the project itself. For example, the Apache Web Server project hosts not only source and binary code, but also documentation, mail lists, frequently asked questions (FAQ) lists, and bug reports. Archived mail lists are generally a good place to start researching any problem, and many projects (such as Perl) now have mailing lists dedicated to new users. Chances are good that a similar problem has been reported, answered, and archived. If the answer can’t be found in the archived mail list, one is free to post a question. But caution is advised—most OSS communities are reluctant to spoon-feed answers to individuals who haven’t spent the time and effort researching the answer on their own. Thus, it’s advisable that IT staff be familiar with relevant OSS projects, and also, perhaps, spend time monitoring relevant mail lists and discussion groups.

Sometimes the move from proprietary products to OSS is too great a leap and a customer may prefer the safety net of a proprietary software vendor or system integrator. Several companies, including vendors such as Red Hat, Novell, JBoss, and MySQL AB, provide support and consulting services. System integrators, such as IBM, HP, and many others, are happy to provide support contracts for OSS solutions.

But it’s important to note that these support and consulting services come at a considerable price, often as much or more than the proprietary software alternatives, especially if a customer chooses to enhance or customize their OSS solution. Not all open source packages have enterprise-class support providers, nor are there strong support commitments from the vendors that package OSS packages as part of a proprietary product. For example, both Red Hat and Novell include over 700 open source packages in their enterprise Linux distributions, but neither company will commit fully to supporting all packages included in their distribution. For additional details on Linux and open source support resources, see the *Application Platform Strategies*

report, “The Enterprise Linux Ecosystem: Free Software, Cheap Hardware, and Expensive Support.”

This has given rise to a new group of companies that have a novel approach for support. Applications built upon OSS often depend upon multiple packages deployed in conjunction as a core application stack. For example, the term LAMP has been coined to describe applications built upon the combination of Linux, Apache, MySQL, and PHP/Perl/Python. Rather than trying to support multiple packages that are updated on different timetables, new startups such as Gluecode Software (now part of IBM), SpikeSource, and OpenLogic, will collect, package, test, certify, and support groups of open source packages that are commonly used in conjunction with each other. Thus, customers relying on these solutions may trade some flexibility (namely timeliness of updates and package selection) for an open source application stack that is tested, supported, and regularly updated.

OSS projects share the same viability and stability risks as proprietary software projects. OSS projects are built and maintained by a community of developers and interested users. As in any community, internal rifts and problems will develop from time to time. At a minimum, this may delay development as the community resolves their differences. Occasionally, these rifts may split the community into separate groups or may cause the community to collapse altogether, either stopping new development or shifting resources to another community.

For example, JBoss, an open source Java 2 Platform, Enterprise Edition (J2EE) application server, experienced a rift between the project lead and several of its key developers, who left in June 2003 to form a separate consulting company. At the time, this split cast a shadow of doubt across the JBoss project, although it has since recovered and gained considerable momentum.

As another example, consider the recent split within the XFree86 community and the creation of X.org. Several XFree86 developers disagreed over licensing changes and were frustrated at the pace at which patches, fixes, and enhancements were released by the XFree86 group. Some of the developers split from the XFree86 group, using the XFree86 source code to create an alternative project, known as X.org, to address licensing issues and community responsiveness. As a result, almost all of the Linux distributions have dumped XFree86 for X.org. Neither of these splits ended development, but both temporarily impacted the development and user community until the rift was resolved.

These rifts occur within proprietary projects as well but are rarely seen by the user community. The open nature and process of OSS gives more insight into internal workings. And because the source is available, forking the project into multiple projects is always an alternative.

OSS project viability is also a concern. While a project released under an open source license will always remain open, future developments may not. For example, SourceForge, an OSS collaboration portal for hosting OSS projects, changed from an open source licensed package to a closed and proprietary product. The core contributors focused future development on the proprietary product, while the open source project never attracted a strong enough developer base. Thus, even though the

source will always remain open, future development may shift to a proprietary product, as was the case with SourceForge.

New Software Licenses

OSS introduces new software licenses to the enterprise, and although there are hundreds of OSS licenses, three are of particular interest: GNU General Public License (GNU GPL, commonly referred to as GPL), GNU Lesser GPL (GNU LGPL, commonly referred to as LGPL), and a set of licenses based on the Apache/MIT/Berkeley Software Distribution (BSD) licenses, which are often grouped together as they're very similar. The vast majority of all OSS software is released under one of these licenses, or a license that has been derived from these core licenses.

The GPL states that any modifications or derived works must also be released under the GPL if the modified or derived work is released to an external party. Thus, if a developer modifies GPL source code and releases the work to the public or to a business partner (either as open source or a proprietary product), the modifications must be given back to the community and the derived work must also be licensed under the GPL. The GPL, by design, keeps the original work and any modifications or derived works as GPL. This is sometimes referred to as the "viral" effect because anything that touches GPL code becomes "infected" and thus must be released under the GPL. Developers rely on the GPL to ensure that their software remains open and free to all.

Note that this is the simplest definition of the GPL; the complete GPL can be found on the FSF's website. The GPL is one of the most popular OSS licenses; the Linux kernel and many GNU development tools, for example, are released under the GPL.

Given the GPL's restrictions, some developers sought a less restrictive license for applications that would benefit from some protections of the GPL without imposing the GPL on the final application. Hence, the LGPL, previously known as the Library GPL, was born. The LGPL allows an application to use LGPL software (commonly libraries) through published interfaces without requiring the complete application to be released under the GPL or LGPL. Simply stated, a program that dynamically links to GPL libraries (such as a Java application) is subject to the GPL. The LGPL permits integration through standard interfaces and dynamically linked programs without imposing either the GPL or the LGPL on the final work. However, any modifications inside the LGPL-licensed work are subject to the LGPL. JBoss is released under the LGPL, thus allowing developers to leverage the JBoss server without licensing their applications under the GPL or LGPL. However, if the developer enhances or modifies the JBoss internal source code, those modifications must be submitted back to the JBoss project. See Figure 1 for more information regarding the GPL and the LGPL.

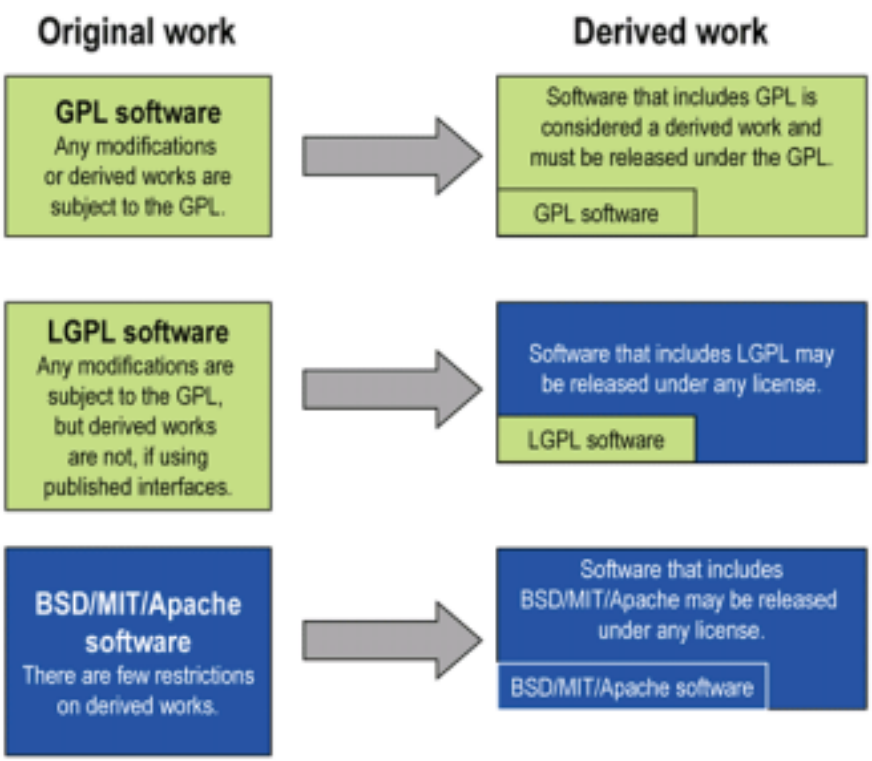

Figure 1: *Common OSS Licenses and Derived Works*

The GPL and LGPL give developers reasonable assurance that someone won't steal their intellectual property, or profit from their work without contributing back to the community. However, sometimes developers want others, especially proprietary hardware and software developers, to take their software without any restrictions.

The second group of OSS licenses—including the original BSD license and the MIT and Apache derivatives—permit anyone to use licensed software without the restrictions found in the GPL and LGPL. For example, anyone is free to take the Apache HTTP server source code, modify it as desired, change the name to BigBob's Web Server, and charge a million dollars per copy. These licenses impose few restrictions; the Apache Software License (ASL) requires only an Apache copyright notice, a list of conditions, and a disclaimer, and its restrictions that prohibit the use of the Apache name for endorsement or promotion of the derived product.

Software licenses can be complex, and the GPL can be particularly perplexing. A great resource to help understand the GPL and LGPL is the FSF's GPL FAQ.

OSS Introduces New Legal Risks

OSS licenses bring new challenges to the enterprise. The primary risks are contamination, derived works, and indemnification. It's important to note that these risks apply equally to open and closed source software projects. However, enterprises are more likely to encounter these problems with OSS due to the ease with which

developers can access the OSS source code.

The first risk, contamination, is minor but worthy of mention. Contamination occurs when a developer moves between open and closed source code bases. Contamination may occur if a developer examines OSS and intentionally or unintentionally uses that information in the design or implementation of their proprietary software. Likewise, a proprietary developer may intentionally or unintentionally share proprietary code with an open source project. In fact, this is the basis of SCO's lawsuit against IBM, which alleges that IBM contributed source code, licensed from SCO, to the Linux kernel.

Independent software vendors (ISVs) are at particular risk because they create software for resale. Thus, there's a chance that GPL-covered source code or concepts may be used in the creation of a non-GPL software product. Note that contamination is difficult to prove, but it's of sufficient concern for ISVs that both Microsoft and IBM have taken specific procedural steps to prevent contamination in their software development process. Users, too, should be concerned if OSS is used to build mission-critical internal applications since these applications may find their way outside the company, either as proprietary projects, or perhaps as connectors or components that enable collaboration and integration with customers and suppliers.

The second risk is that of software derived from or dependent upon GPL software. In the simplest case, a derived work is created when a developer uses GPL code to create a new software product. So, for example, if a developer uses a GPL filesystem as the core of a new and improved filesystem, the new filesystem would be considered a derived work and subject to the GPL.

A more likely scenario is the case of aggregation, in which a proprietary application is combined with or depends upon a GPL application. In some cases, the entire work may be subject to the GPL.

To distinguish "combining" from "aggregating," the FSF's FAQ about GPL states:

> Combining two modules means connecting them together so that they form a single larger program. If either part is covered by the GPL, the whole combination must also be released under the GPL—if you can't, or won't, do that, you may not combine them.What constitutes combining two parts into one program? This is a legal question, which ultimately judges will decide. We believe that a proper criterion depends both on the mechanism of communication (exec, pipes, rpc, function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are interchanged).If the modules are included in the same executable file, they are definitely combined in one program. If modules are designed to run linked together in a shared address space, that almost surely means combining them into one program.By contrast, pipes, sockets and command-line arguments are communication mechanisms normally used between two separate programs. So when they are used for communication, the modules normally are separate programs. But if the semantics of the communication are intimate enough, exchanging complex internal data structures, that too could be a basis to consider the two parts as

combined into a larger program.[1]

For example, under strict interpretation, a developer using a GPL Java Class in their application must release their entire application under the GPL if they distribute their application externally.

The third and most substantial risk is the lack of indemnification for users from vendor patent and copyright disputes, a protection offered by most proprietary software licenses. For example, when Company A sells an office suite to a user, it will indemnify that user from any lawsuits brought against Company A for patent or copyright infringement. Thus, if Company A is found guilty of a copyright violation, Company A is responsible for damages, not the users of its product.

Current OSS licenses have no indemnification clause. After all, how can a nonprofit OSS community agree to insulate users from copyright and patent disputes, especially over code that may be developed by thousands of individuals from around the globe? Without indemnification, if someone contributes copyrighted code to an OSS project, users of that project are liable for damages.

In May 2003, SCO sent letters to 1,500 of the world's largest Linux users. SCO believes that some of its UNIX copyrighted code was released by IBM into the Linux kernel. Thus, due to the lack of an indemnity clause in the GPL, SCO believes it is within its rights to pursue users for license fees for its intellectual property included in the Linux kernel.

Whether or not SCO succeeds is still being determined by the courts, but SCO's case against IBM is not looking strong. However, SCO's actions validate the immaturity and risk of OSS licenses. Some OSS vendors, systems integrators, and even insurance companies will provide indemnity for OSS packages. Today, perhaps spurred by SCO's actions, many companies provide indemnification for OSS. Examples include HP, JBoss, Novell, and Sun. There are specific restrictions and limitations, but proprietary software vendors with a vested interest in OSS are stepping up to provide protection to their customers.

This has also given rise to new businesses that scan for potential OSS infringement or OSS licensing conflicts. Black Duck Software and LogicLibrary provide tools and resources for scanning internal software projects and comparing them against known open source projects. These tools typically work by creating hashes of internal and open source projects; compliance checking doesn't require actual exchange of source code, just comparison of hash values. Black Duck Software even provides online services (protexIP/OnDemand) for limited use scans of source code, such as what may be required for due diligence during a licensing or acquisition.

Because most OSS licenses have not been tested in the world's courts, it's difficult to determine an enterprise's actual risk when using OSS. Likewise, it's difficult for a vendor, system integrator, or insurance company to fairly price indemnity insurance as there's no court or damage precedent to use in balancing risk against insurance cost. The need for indemnification arises when an OSS contributor places proprietary intellectual property into an OSS product. Can the courts really hold OSS users

responsible for every line of source code in their OSS-based deployments?

There's also another considerable gray area regarding websites and web services built upon GPL software. Today, a company is free to modify GPL software and keep those changes private as long as the modified code is never released to partners or the public. However, this may change with future versions of the GPL. The FSF's GPL FAQ sheds a bit of light on current thinking regarding websites and web services built upon the modified GPL software:

> **A company is running a modified version of a GPL'ed program on a web site. Does the GPL say they must release their modified sources?**
>
> > The GPL permits anyone to make a modified version and use it without ever distributing it to others. What this company is doing is a special case of that. Therefore, the company does not have to release the modified sources.It is essential for people to have the freedom to make modifications and use them privately, without ever publishing those modifications. However, putting the program on a server machine for the public to talk to is hardly "private" use, so it would be legitimate to require release of the source code in that special case. We are thinking about doing something like this in GPL version 3, but we don't have precise wording in mind yet.[1]

As previously stated, many of these risks aren't limited to OSS. In fact, most hypothetical risk scenarios for OSS apply to an enterprise that has licensed proprietary software binaries and source code.

However, the internal processes and procedures used when acquiring OSS versus proprietary products are different. For example, most enterprises will carefully review proprietary software licenses, generally by an internal legal team or staff familiar with contracts, and will take special caution in managing proprietary licensed software. OSS is extremely easy to acquire and is available to everyone, and this means that developers or IT staff may make software license choices for the organization without fully understanding the open source license or potential consequences. It's not uncommon for a development group to believe that open source is "public domain" or to draw incorrect conclusions about what's allowed or not allowed by the software license. Complicating matters is the fact that there are dozens of open source licenses, some with significant implications.

Software Commoditization—Full Speed Ahead

As Burton Group examines open source initiatives, an underlying theme is emerging: "good enough." OSS doesn't have to be the best or most innovative solution; rather, it must be good enough for the task at hand while delivering a significant benefit over rival or incumbent solutions. Often, the benefit is price: An Apache HTTP server on Linux is good enough for serving webpages but less expensive than a proprietary UNIX-based hardware/software solution. But it's not just cost—the significant benefit may be time to

market, interoperability, or freedoms uniquely found with OSS. For example, Eclipse is quickly capturing market share and ISV attention within the integrated development environment (IDE) landscape. Eclipse wins because of the Eclipse ecosystem. Many ISVs are now writing Eclipse plug-ins rather than developing their own IDE. These ISVs leverage a strong Eclipse code base and the community of developers already familiar with Eclipse.

Good-enough solutions act as market disruptors, which may (or may not) lead to commoditization. Many OSS projects are disruptors: they're not necessarily as good as the incumbent product or technology, but serve a narrow problem space that's of vital concern to the customer. For example, some customers are choosing to combine a servlet engine (Tomcat) with a persistence layer (Hibernate) in lieu of a J2EE deployment. Another form of disruption is displacement, when a technology nullifies the necessity of another product.

Commoditization occurs when products become undifferentiated so that lower cost brands cannot be easily differentiated from high-cost brands. In this case, the price is driven down because there is no benefit to choosing the more expensive brand. This, in turn, places considerable pressure on the market incumbents. Open source, for some markets, is a disruptor that may (but not always) lead to commoditization. For example, Apache has commoditized the web server market, providing a good-enough alternative to commercial products. Open source database management products, such as MySQL and PostgreSQL, are disruptors, but the database management system (DBMS) market is not commoditized.

Commoditization isn't a recent phenomenon, and isn't directly related to open source technologies. For example, Microsoft has pressured incumbents by introducing products that were not necessarily innovative or leading-edge technology—they were sufficient for the task at hand but delivered a 10x savings, either through pricing or packaging, with no substantial differentiation. Consider the case of the Transmission Control Protocol/Internet Protocol (TCP/IP) vendors in 1994: Prior to Windows 95, there was a market for TCP/IP protocol stacks and utilities. With Windows 95, Microsoft introduced good-enough alternatives—not leading edge, but sufficient for most tasks, with additional benefits such as being free (included in the price of the operating system) and integrated with the operating system and other services.

Fast forward to 2005, and OSS can be seen to have the same effect. Before OSS enters a software market, that market determines product features and price points. Competitors balance feature cost against product price, trying to find a profitable market sweet spot. OSS upsets this balance. With OSS, software licensing costs are zero, although implementation and support costs remain (or may increase). Vendors face pricing pressures when consumers have good-enough alternatives available via open source projects. Linux and Apache are great examples of OSS commoditizing the operating system and web server markets.

Likewise, it may be difficult for a proprietary vendor to compete against OSS projects on a feature-by-feature basis. Features and functions similar to those that are built at a cost by a vendor can become part of the OSS project at no cost to the consumer. In addition, it may be harder for vendors to stratify their own products based on features, such as limiting connections or database size, since competing OSS projects impose no such

restrictions. Vendors may become increasingly hostile toward OSS projects that duplicate the very features they're spending resources to create. Hence, it may spark a new round of intellectual property lawsuits aimed at OSS vendors and consumers, as proprietary vendors see their features delivered for free in OSS projects.

There's no guarantee that OSS will commoditize any given market segment. To date, OSS has been more successful commoditizing infrastructure components with large markets, as seen in the operating system and web server markets. OSS is less successful higher in the solution stack, as seen in Figure 2, although it is making some interesting inroads. In general, OSS will tend to commoditize those black-box services, where the input and output are well known or defined through open standards.

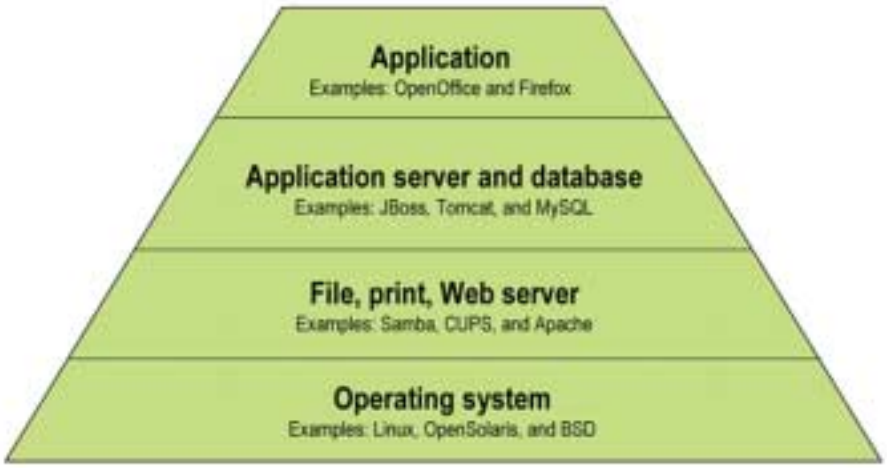

Figure 2: *OSS Is Commoditizing All Layers, but Its Presence Is More Extensive at the Lower Infrastructure Layers*

OSS Business Models

OSS is a threat to some vendors but a boon to others. It's changing the business of giants such as IBM, Microsoft, Novell, and Sun, while offering new opportunities to startups such as MySQL and JBoss. OSS delivers new options while exerting pricing pressures on market incumbents.

There are several examples of OSS causing a market incumbent to change product pricing or packaging. For example, Microsoft offers Windows 2003 Server Web Edition for less than $399 to target the low-end web server market in which Linux and Apache have been quite successful. Microsoft also offers the Microsoft SQL Server 2000 Desktop Engine (MSDE 2000), a free, redistributable version of SQL Server for client applications. Microsoft has also been much more liberal in providing evaluation software, making it almost as easy to acquire Microsoft's products as it is to acquire open source products. Macromedia recently announced a non-proprietary deployment license for its Flex server, which was driven, in part, by a competitor (Laszlo) releasing its presentation server under an open source license as OpenLaszlo.

Which vendors will survive the disruptive effects of OSS? To understand the impact of

OSS on your vendors, it's helpful to understand how they will compete in an OSS world.

Hardware

Hardware vendors generally favor OSS because it doesn't directly compete with their business. Unlike software, hardware costs will never go to zero, yet open source often lowers the cost of delivering hardware-based solutions. For example, OSS has enabled a new set of low-cost appliances and embedded devices, such as proxy-cache servers, firewalls, routers, and network attached storage (NAS) solutions. Rather than investing internal resources for developing embedded operating or management systems, hardware device manufacturers can leverage an extensive library of open source projects, which fast-paths their time to market and reduces development costs. At the same time, because these hardware systems leverage open source projects, it draws a community to the product line. One great example is Linksys' line of wireless routers and access points built upon Linux. Several open source projects and proprietary vendors now offer enhanced firmware for these devices, and this, in turn, has increased demand for the hardware. OSS also enables hardware vendors to repurpose existing hardware into new markets or extend the life in current markets, such as IBM's zSeries mainframe, modified to run multiple Linux virtual machines.

In particular, OSS has been a great advantage to the commodity Intel-based server vendors. OSS in general (and Linux in particular) gave IBM, HP, and Dell the necessary hardware and operating system solution to replace proprietary UNIX servers (mainly Sun). Consider this impact for a moment—IBM, HP, and Dell, along with Microsoft, tried for years to replace Sun's proprietary hardware and software with commodity x86 hardware and a Microsoft operating system. They found, however, that switching customers to an unfamiliar hardware and operating system was too difficult. Then, along came Linux on Intel, a solution that was proven to be good enough for many Solaris/SPARC functions. More important, however, is that because Linux was close enough to Solaris, switching from Sun to Linux/Intel was easier than switching to Windows/Intel and required little IT retraining. As previously mentioned, almost every customer interviewed regarding Linux cites low-cost Intel hardware as the primary driver toward Linux. For additional details on Linux and open source support resources, see the *Application Platform Strategies* report, "The Enterprise Linux Ecosystem: Free Software, Cheap Hardware, and Expensive Support."

Services and Support

There's a sustainable market in the service and support of OSS solutions, just as there's a market for service and support of proprietary products. But OSS actually produces new opportunities for consulting firms and system integrators beyond those that are possible with proprietary software products.

Two new primary market segments have been opened within the OSS service and support arena. The first segment includes vendors that specialize in packaging, testing, and maintaining OSS solutions. The second includes system integrators that specialize in installing, customizing, and integrating OSS solutions.

For example, consider the leading proprietary Linux distributions. Novell and Red Hat pull hundreds or thousands of OSS packages together, along with some in-house applications (such as an installer), into a Linux distribution that's packaged, tested,

certified, and supported. Novell and Red Hat fill a gap required by enterprises—namely, service and support, with long-term commitments to support Linux versions for many years, rather than the rapid update and end-of-life time frame that occurs in hobbyist Linux versions.

Because the GPL requires giving away all of the source files with the Linux distribution, these vendors have business models that are sustained by selling subscriptions to patches and updates, and by providing additional service and support. Both companies comply with the GPL by providing the source code to their enterprise Linux products, but they do not provide a complete distribution of compiled binaries. In theory, anyone is free to create competing enterprise Linux products by pulling from the same sources, and that's exactly what CentOS, Tao Linux, and White Box Linux deliver. But the enterprise Linux vendors are betting that the complete package—software, integration, testing, support, and updates—is less expensive for customers than building and maintaining their own distribution.

OSS also creates new opportunities for system integrators. First, system integrators can use free OSS software instead of proprietary components, to either reduce customer costs or increase system integrator profits. Second, system integrators can pull from a multitude of OSS projects to create best-of-breed solutions, even solutions that mix OSS and proprietary products. Third, access to source code enables system integrators to customize or enhance OSS components, something that's difficult or impossible with proprietary software components. Finally, from an accounting perspective, it may be easier to absorb a custom software solution based on OSS than proprietary software. For example, consider two projects valued at $1 million each: one project that's half proprietary software (a capital expense) and half consulting (an operating expense), and a second project that's OSS (no capital expense) and all consulting. For businesses that wish to avoid capital expenses, OSS-based projects may be more attractive because they're recognized as operating expenses and have certain tax advantages.

IBM, Computer Associates, EDS, and Red Hat are growing their OSS custom development and integration services. IBM, for example, provides several Linux and OSS consulting offerings, including an OSS offering that covers topics such as licensing, technical, legal, export, and ethical issues associated with OSS projects. For many system integrators, OSS is not a competitive threat, but a welcome new environment for selling their services.

Management and Integration

One of OSS' greatest strengths is its ability to rally a community around a technical issue or problem. Need a better file-locking mechanism or journaled filesystem? How about an instant messaging client that connects to multiple providers? There are good OSS solutions for these point problems. However, one area in which OSS is particularly weak is integrated solutions that span multiple OSS projects.

Consider Windows—all of the desktop services are highly integrated and tightly coupled. Microsoft has done a good job of integrating several point technologies into a complete turnkey solution. The various desktop systems and applications gain greater value from Microsoft's efforts to integrate point technologies into a complete solution.

Although OSS is great at the point technologies, it's weaker at the integrated solution, but that gap is being addressed by the proprietary vendors. The developers of many popular projects, such as Apache, PHP, and MySQL, are aware that their solutions will be used in conjunction with each other, but often the integration is an exercise left to the user. Some companies, such as the Linux vendors, provide additional integration between various OSS projects, but not to the extent found in proprietary software products. Still others, such as Gluecode, OpenLogic, and SpikeSource, offer an integrated stack in which configuration wizards take inputs for the complete project, and then individually configure a set of open source packages. Lack of integration is a double-edged sword. On one hand, it increases administrative tasks and configuration, or requires additional add-ons or products. On the other hand, by not assuming or mandating commonality between the components, developers and system integrators have more flexibility in producing custom solutions. This is, as Eric S. Raymond describes in his "The Cathedral and the Bazaar," a bazaar of software components where developers can pick and choose as required by their projects, but where they're also shouldering much of the integration effort.

Dual Licensing

"Dual licensing" is a term applied to OSS projects released under multiple licensees that allow both proprietary and non-proprietary use. For example, MySQL, a popular OSS DBMS, is released under the GPL. Enterprises deploying internal applications requiring a DBMS or developers building GPL applications can use MySQL under the terms of the GPL. But other users, such as commercial ISVs, may not be able to use GPL software in their commercial product. For these users MySQL also licenses the MySQL DBMS under a commercial license. Dual licensing enables MySQL to work with the OSS community, and to profit or recoup development costs from proprietary entities using MySQL. For MySQL, most of the community benefit comes from testing, not code contributions. Even so, testing is a considerable software effort; by releasing MySQL under the GPL, MySQL can enlist tens of thousands to test and provide feedback. Thus, MySQL's relationship with the open source community is dependent more on testing than code contribution.

Upon initial inspection, dual licensing may seem to violate the GPL. After all, how can a product such as MySQL that's released under the GPL also be sold as part of a proprietary offering? To answer this question, one must remember that copyright holders are within their rights to determine the licensing for their software, and are free to release their software under more than one license. Thus, a vendor may choose to use an open source license for some types of users and uses, and a proprietary license for other situations. Many other OSS projects are dual licensed: Trolltech's Qt, a C++ application framework, has used dual licenses (GPL and proprietary) since 1994; OpenOffice is offered under the GPL and Sun Industry Standards Source License; and Sleepycat (another open source database) is also dual licensed.

Dual licensing is a viable model that gives the OSS vendor the benefits of the OSS community and the ability to recover some costs when its software is used in proprietary products. But, as with most proprietary versions of open source projects, prices are kept in check because there are always free alternatives readily available. And that's good for consumers.

Application and Data Lock-In

There are many capable OSS alternatives to proprietary software products, but migrating to these alternatives may incur high switching costs and compatibility risks. Consider the amount of data in Windows NTFS filesystems, Oracle databases, and Windows Office file formats. Moving this data to an OSS alternative is not straightforward and often requires human intervention to ensure that nothing is lost in the conversion. After all, moving files is trivial, but moving permissions, symbolic links, and data relationships is not a trivial task. But it's not just data—consider the number of applications running on Win32 application programming interfaces (APIs) or other proprietary application formats. If data migration is tricky, applications are even more difficult.

The OSS community is working hard to ensure interoperability with proprietary data formats, but the effort is often stymied by the incumbent vendor who holds the data. In some cases, this data lock-in may be the primary reason a customer has not migrated to an OSS alternative. Even 95% or greater compatibility may not be acceptable to users, especially when data must be exchanged with external customers. One incompatibility may result in a lost customer or help desk call, thereby eliminating any OSS cost savings.

Application integration, and eventual migration, is getting easier due to the advances and proliferation of web services technologies. Proprietary application vendors are adopting the Extensible Markup Language (XML)-based formats and protocols that provide greater interoperability between dissimilar systems. As a result, over time it will be easier to wrap proprietary applications with web services technologies. Eventually, this moves the application to a black-box device, thereby simplifying replacement with an OSS technology that maintains the same XML formats, functions, and protocols of the proprietary system. Note that this won't be a simple conversion; XML web services, while maturing rapidly, must address other key areas before there will be widespread adoption. For more information, see the *Application Platform Strategies* overview, "The Advent of the Network Platform: Web Services Move into the IT Fabric."

Given the application and data lock-in issues, several vendors will enjoy many years of continued revenues even though there are viable OSS challengers. Although lock-in is a risky survival strategy for vendors, it can be an important part of a total solution that augments the lock-in with support, indemnification, and continued feature improvement. As Burton Group has recommended, proprietary vendors must strive to fill the gaps overlooked by the OSS world. The application and data incumbents are in a strong position to resist OSS commoditization.

It's the Intellectual Property, Not the Software

OSS is most successful at commoditizing the infrastructure layer, in which features and functions are well defined by standards, such as those for file, print, web server, Java/J2EE application server, mail, and so forth. In these infrastructure markets, OSS is exerting pressure on the incumbent proprietary vendors whose services are delivered as black-box devices in which a given set of inputs returns outputs, as specified by a standard.

However, OSS will have a much harder time replacing incumbents in several software

market segments. The role of software changes as one moves up the stack from infrastructure to applications and specific business domains. Software becomes the vehicle for delivering unique intellectual property. For example, consider the case of a tax software program. The value of the software isn't in its data-entry windows and its ability to print tax forms, but rather in its intricate understanding of complex local, state, and federal tax laws. Software then becomes the vehicle for delivering that specialized intellectual property. OSS is good at commoditizing the underlying infrastructure components, but it can't similarly handle the intellectual property that's part of the tax software package.

That said, OSS will impact these software markets. Primarily, it will allow developers of these applications to leverage the OSS commodity infrastructure when building their software. Rather than license runtime application foundations from proprietary vendors, these developers can turn to a wide variety of OSS components, so long as the OSS components have compatible licenses that permit incorporation into proprietary software products. Theoretically, as the cost of developing applications decreases when using commodity OSS components, these savings should be passed on to the consumers.

Some consumers are already seeing these benefits. Several Java application server vendors are leveraging commodity OSS components, such as Apache Axis, in their proprietary software products. IBM's recent purchase of Gluecode brings IBM's service and support momentum behind the Apache Geronimo open source J2EE server. This, combined with the commoditization of the infrastructure (again, due in part to OSS), has reduced the prices of many proprietary Java application servers. For additional information on the threat that open source brings to the J2EE marketplace, see the *Application Platform Strategies* report, "J2EE: A Standard in Jeopardy," and the *Application Platform Strategies* overview, "The Rebel Frameworks: J2EE Open Source Alternatives and Supplements."

Sue Your Customers and Competitors

SCO is testing the waters regarding intellectual property and OSS. SCO's challenge consists of two parts. The first is a lawsuit against IBM, alleging that IBM placed SCO-copyrighted UNIX source code and intellectual property into the Linux kernel. The second is an attempt to convince software vendors and users that SCO is entitled to software royalties for the use of Linux or other UNIX-related technologies.

SCO alleges that thousands of lines of UNIX source code were directly copied into Linux under IBM auspices, and were then released under the GPL. SCO has been reluctant to release details of the infringing sections of Linux, either by requiring a nondisclosure agreement (NDA) or obfuscating the source code. Many in the Linux community believe that once SCO shows the offending code, it will quickly be replaced with clean code, not subject to SCO's allegations. Others have pointed out that SCO also released the offending source code under GPL with its Caldera Linux product, thus negating any copyright issues.

It's unlikely that SCO will win against IBM. SCO is quickly running out of funds to continue its legal pursuit of IBM. Yet SCO's actions have sparked discussion among customers regarding intellectual property and indemnification. Even if SCO fails, others may follow. Microsoft, in particular, is adamant about protecting its intellectual property,

and has the means, desire, and patent portfolio to prevent OSS from further encroaching on its lucrative businesses. Will other vendors follow when they see their intellectual property in OSS projects?

Vendor Reactions

OSS affects hardware, software, and services. Vendor reactions across these market segments are mixed. Some, such as IBM and Oracle, are rapidly incorporating OSS into their businesses. Others, such as Sun, have sent mixed messages to users and the OSS community. While there are too many vendors to be analyzed in detail, the following vendors stand to be most affected by OSS adoption.

IBM

IBM was an early OSS adopter and continues to be both a driving and positive force in the OSS community. IBM has claimed that in 2001 it invested over $1 billion in Linux and recouped most of those costs by 2002. IBM plans to invest $100 million over the next three years in Linux and other OSS technologies just for the IBM Workplace collaboration portfolio, which includes Lotus Notes, Domino, and WebSphere Portal. IBM also claims to have over 7,500 employees researching, developing, selling, and marketing Linux. But, IBM's investments and contributions to the OSS community don't stop there; it currently hosts or participates in more than 162 OSS projects. While some projects are legacy or a bit self-serving, many projects are valuable contributions to the OSS community, including substantial contributions such as the Eclipse integrated development platform. IBM also participates in several Linux kernel and filesystem projects, the Samba file server, many Apache projects, and PHP.

Some OSS projects compete with IBM's software business, such as Linux's potential replacement of AIX, but the majority of OSS projects benefit IBM's hardware and services business. For example, through OSS, IBM has breathed new life into its zSeries mainframe business. Porting Linux to the zSeries mainframes required IBM to work with, not against, the OSS community. And, like many other proprietary software developers, IBM benefits from leveraging the OSS infrastructure components in its software line.

IBM, unlike other vendors profiled in this section, has strongly embraced OSS as a core component of its business. As evidence of IBM's commitment, consider the recent acquisition of Gluecode, an integrated J2EE application server stack built upon Apache Geronimo plus additional open source projects for portal, database, and messaging services. This acquisition marks a clear open source commitment by IBM, even though Gluecode SE will compete with IBM's own application server offerings.

Ironically, the big loser in this acquisition may be JBoss, not IBM's own WebSphere application server. JBoss has considerable market share and momentum, and has been a powerful force in commoditizing and driving down prices for J2EE applications servers. Apache Geronimo, while not nearly as mature or complete as JBoss' open source product offering, is released under a license (the Apache license) that's much more attractive to ISVs, original equipment manufacturers (OEMs), and even some customers. With this acquisition, IBM is clearly backing competing alternatives, while at the same time building a bridge from the low-end or commodity J2EE application sever

into the IBM WebSphere superplatform.

For additional details on market dynamics and product offerings, see the *Application Platform Strategies* document series, “J2EE: A Standard in Jeopardy” and “The Java Superplatforms: Comparing IBM, Oracle, and BEA,” and the *Application Platform Strategies* overview, “The Rebel Frameworks: J2EE Open Source Alternatives and Supplements.”

Sun Microsystems

Sun has sent mixed messages regarding OSS. On one hand, Sun has made enormous contributions to the OSS community—more than any other vendor, including IBM. Sun has given the OSS community projects such as OpenOffice, NetBeans, and NFS. Sun also contributes to several other important OSS projects including Apache Ant, Apache Tomcat, Gnome, JXTA, and WEBM. SunSource (www.sunsource.net) is Sun’s clearinghouse of links and information regarding Sun’s OSS involvement. On the other hand, for all of Sun’s contributions, its good will is sometimes overshadowed by its mixed messages regarding Linux and Java.

Sun’s revenues and profits have been significantly impacted as customers replace Sun servers with Linux on commodity Intel hardware. This has not been a wholesale migration from SPARC and Solaris, but rather a movement of low-risk infrastructure services to Linux, which is both free and good enough for these tasks. However, as Linux and other OSS projects mature, these solutions are becoming good enough for a wider variety of tasks, putting even more pressure on Sun’s core business.

Sun has taken several dramatic steps in response. The latest release of Sun’s operating system, Solaris 10, marks both a turning point and a serious gamble for Sun. Solaris 10 is released as open source under the Common Development and Distribution License (CDDL). Solaris 10 also delivers new core operating system features, a serious commitment to 32- and 64-bit x86 architectures, and integrated open source packages. Taken as a whole, Solaris 10 is clearly aimed at stopping Solaris to Linux migrations. And although the rhetoric is thick from Sun, its Linux competitors, and the Linux community, Burton Group believes that this is not a winner-take-all game. In fact, closer examination of the technology, market drivers, pricing, and customer requirements suggest that Solaris 10 may be a better choice than Linux in some cases, especially for businesses with Solaris experience who depend upon OSS applications and runtime frameworks. For additional information, see the *Application Platform Strategies* report, “Solaris 10: Does Sun Have a Compelling Linux Alternative?”

Sun has also sent mixed messages regarding OSS and Java. The Apache Software Foundation successfully lobbied Sun in early 2002 to open several aspects of the Java Community Process (the process that defines new Java features), but only after a long and public disagreement in which Sun appeared to be anti-OSS. And until recently, Sun has been hesitant to certify open source J2EE implementations such as JBoss and Apache Geronimo.

Sun also continues to resist market pressure to open source Java. Although Sun does occasionally release reference implementations of some standard Java APIs through open source, the core platform remains closed. In March 2005, Sun announced a new

licensing scheme for Java, which will apply to the next release of Java 2 Platform, Standard Edition (J2SE), code-named "Mustang." The new licenses are discussed in the "Sun's New Java Licenses" section of this overview.

Although these new licenses are slightly more open than before, they do not deliver the true benefits of open source. Two of Sun's new Java-related licenses, the Java Research License (JRL) and Java Internal Use License (JIUL), give users access to the source, but they do not give developers the freedom to use the source as they like. A litmus test for open source is that it allows developers to build a new project from existing open source project(s), also known as "forking the code." Sun's new licenses clearly prevent developers from forking the Java code base. Granted, Sun is forthright about these licenses as they clearly stated that these licenses are not open source.

Sun defends its position on open sourcing Java by claiming that only closed source can protect Java compatibility. But this is just an excuse. Sun protects Java compatibility through licensing the Java brand, not through licensing the source code. An implementation may not be called "Java" unless it passes the Test Compatibility Kit (TCK). Using the GPL—or even CDDL (the new Solaris 10 open source license)—Sun can use a dual-licensing strategy to open up the source code and still maintain a proprietary licensing practice. To some, this makes it appear that Sun would like it both ways: It wants the benefits of open source—getting vast hordes of developers to contribute fixes and new ideas to the Java code base for free—but isn't willing to actually contribute the source code to the open source/research community.

Although Sun's licenses do not open Java as would other licenses (like the GPL), they do provide some value to users. One valuable benefit of open source is that it allows users to debug, fix, and tweak code to suit their specific requirements, and the JIUL does give a company that capability. However, few companies actually take advantage of this opportunity. Java is a bit too complex for most companies to want to muck around in. It's also potentially dangerous—because if user companies do muck around in the code, they are likely to break compatibility.

Microsoft

OSS poses a real, long-term threat to Microsoft—a threat that Microsoft takes seriously. Microsoft's largest revenue streams—Windows and Office—are under attack by Linux and OpenOffice. Linux is growing faster than Windows on enterprise and Internet servers, and OpenOffice is good enough for some Office tasks, although compatibility with Office documents remains a challenge. In a strange twist of fate, Microsoft's longstanding hardware partners, who were instrumental in building the Windows desktop and server franchise, are now promoting Linux as an alternative, offering their hardware pre-installed with OSS alternatives. Appliances running open source alternatives, such as NAS devices, are also encroaching on the Windows server market.

OSS is unlike any competitor Microsoft has ever faced—software priced at zero can't be undercut. Microsoft tried to spread fear, uncertainty, and doubt (FUD) about the GPL, but only alienated itself further from its own customers who use both Microsoft and OSS products. The company has even lobbied governments to take action against OSS, believing that the GPL, in particular, will erode the worldwide proprietary software

market.

Microsoft's Shared Source Initiative FAQ states:

> Microsoft does have a concern with the effects of the GPL on the broader software ecosystem, particularly within the governmental and academic research sectors. This ecosystem has sustained unparalleled innovation throughout the industry for the past quarter-century. The principal role of government and universities in the ecosystem is to undertake basic research and to dispense the findings both into the societal base of technical knowledge and to private enterprises and individuals capable of developing these innovations commercially. Commercial enterprises, in turn, engage in applied research to develop products that advance the state of technology—generating jobs, profits, and tax revenues that boost the economy, which then funds additional basic research in the process. Commercial enterprises also disseminate innovations directly into the larger technical-knowledge base.Basic-research technology licensed under the GPL—as contrasted with research technology distributed into the public domain or under other open source licenses such as the FreeBSD license—could not be developed distributed [sic] for direct commercialization. As a result, the technological innovations stemming from public and academic research that frequently accompany commercial product development would not come to pass. The commercial sector would have a more difficult time fulfilling its historical role of refueling the ecosystem with funding and contributions of technical knowledge.Microsoft believes that the combined forces of a well-funded research community and a robust commercial software industry will continue to drive global economic expansion. In the context of basic research, however, the GPL threatens to sever this long and successful partnership.2

Ironically, OSS uses tricks from Microsoft's playbook, combining low-cost (free) solutions with good-enough solutions. For many file, print, web server, and application platform services, OSS projects are free and good enough to displace a Microsoft server. And, just as Windows enjoyed good support from hardware OEMs, OSS (particularly Linux) plays an important role in the business strategies of IBM, Dell, and HP.

But Microsoft is learning quickly. It has taken a softer approach against OSS licenses and the community. For example, Microsoft is now involved at OSS-related activities, sponsoring lunches and events where it can endear itself to the non-Microsoft developer community. Recognizing the value of OSS source code transparency, it has created programs that allow governments, customers, and developers a limited look inside product source code. Microsoft has even released the Windows Installer XML (WiX), a toolset for building Windows installation packages, on SourceForge under the Common Public License.

Open source solutions will have a difficult time undoing Microsoft's lock on desktops and the office productivity market. Servers are easier to replace than desktops because (in theory) their replacement has less direct effect on users, and they host a limited number

of infrastructure applications or functions. However, desktops have a direct user impact. The number of Windows applications dwarfs viable OSS desktop applications alternatives, and Windows applications are typically more mature. In addition, data incompatibility, user retraining, and support issues associated with the desktop migration quickly eliminate any OSS cost savings. Still, several customers are toying with Windows desktop replacements as they consider the other OSS cost savings and advantages, but wholesale migration away from Microsoft Office and the Windows desktop is at least three to five years away, if it ever happens.

Red Hat

Red Hat is one example of a potentially sustainable OSS service business. Although many consider Red Hat a software company, revenues are generated by software subscription and enterprise services. Red Hat understands that it's difficult to compete with OSS and no longer bases its business on a retail software product.

The company has found a successful business in the maintenance of Red Hat Linux distributions, selling not a software package, but rather a subscription to the latest enhancements and updates, tested and integrated by Red Hat. And enterprise customers are subscribing to the service because many find it's less expensive to depend on updates and fixes from Red Hat's subscription service than it is to duplicate these services in-house. Subscriptions grew 38% in the 2004 fiscal year, contributing $124.7 million in revenue. Red Hat now has more than 200,000 Red Hat Enterprise Linux subscribers.

However, this subscription model isn't foolproof, as explained in Red Hat's 2004 10-K filing with the Securities and Exchange Commission:

> The subscription agreement for Red Hat Enterprise Linux requires customers to agree to a subscription for our systems management services for each machine on which they deploy Red Hat Enterprise Linux. At the same time, the subscription agreement places no restriction on the customer's right to redistribute Red Hat Enterprise Linux. While we believe this practice fully complies with the requirements of the GNU General Public License, and while we have reviewed this practice with the Free Software Foundation, the organization that maintains and provides interpretations of the GNU General Public license, we may still encounter customer resistance to this distribution model. To the extent we are unsuccessful in promoting or defending this distribution model, our business and operating results could be materially and adversely affected.[3]

Thus, customers are free to install Red Hat Enterprise Linux on any system without paying a subscription to Red Hat. These systems are not entitled to patches, updates, and support, although the GPL does guarantee that any source code for patches and updates must be made available separate from the subscription service. Red Hat must find a price point that's attractive enough for enterprise customers (as compared with a proprietary software version) but not so high that it inspires an OSS project that competes with their subscription service. To date, at least three free or low-cost "generic" versions of Red Hat Enterprise Linux are available: CentOS (www.centos.org),

Tao Linux (http://www.taolinux.org/), and White Box Enterprise Linux (www.whiteboxlinux.org).

Oracle

While other vendors are embroiled in public OSS controversy, Oracle is behind the scenes, adopting and using OSS technologies, but only when it's convenient and not threatening to its core market. On one hand, Oracle must adopt OSS for its survival. For many years, Oracle and UNIX were inseparable twins in the corporate data center, but Linux has changed this environment, displacing some UNIX servers typically deployed to host Oracle databases and applications. Oracle has a long history of multiplatform support, so it's no surprise that it was willing to adopt Linux as yet another platform for its products. At the same time, Oracle realized that there were shortcomings in the Linux operating system and filesystems; as such, it contributed considerable resources and source code to improve Linux distributions, thus ensuring that Oracle could thrive on the Linux platform Oracle now offers its database, development tools, application servers, and services on Linux and other OSS technologies. Oracle has also adopted other OSS technologies, such as the Apache Struts Web Application Framework, because it helps developers build applications on Oracle's application server.

On the other hand, Oracle has not embraced any open source projects that compete with its core products. Unlike IBM, Oracle has not embraced any DBMS or application server open source projects.

Open source DBMS projects, such as PostgreSQL and MySQL, and application server projects, such as JBoss and Gluecode, compete with Oracle's core business. These projects provide commoditized, good-enough solutions that chip away at Oracle's low-end business. In response, Oracle has focused on moving up the stack, providing differentiated solutions with features such as high availability, scalability, management, security, and integration. At the same time, Oracle is advancing up the stack by acquiring companies such as PeopleSoft. Likewise, there's an enormous amount of data and number of applications built on Oracle's technologies; switching to an OSS solution, in many cases, is technically impossible or cost prohibitive. And Oracle deployments are often mission critical where there is a need for the service, support, and service level agreements (SLAs) that are missing from OSS alternatives.

Oracle, like other vendors, has implemented creative product packaging and pricing to combat popular OSS products, such as MySQL and JBoss. For example, Oracle Standard Edition One is a low-cost ($4,995) simplified version of Oracle Application Server 10g and includes database, portal, and J2EE application server capabilities. It has almost all of the same features as the Oracle 10g, but imposes deployment limits such as operating system version (Windows or Linux) and hardware capabilities (a maximum of two CPUs). Oracle indicates that it's intended for the small and medium-size business market, but it is also targeting IT departmental applications and the ISV/OEM market. It's clearly a competitor to open source alternatives, and while not free, is a whole lot richer than good-enough open source alternatives.

Still, OSS momentum is relentless. Oracle must strike a balance between OSS technology, Oracle application and data lock-in, and filling the gaps in OSS solutions. To date, Oracle has made some contributions to the OSS community (mainly in the Linux

kernel and filesystems), but is largely an OSS consumer, not a contributor like Sun or IBM.

The Superplatform Effect

In the *Application Platform Strategies* report, "The Java Superplatforms: Comparing IBM, Oracle, and BEA," Burton Group discussed the emergence of a "superplatform," a highly integrated suite of services built around an application server. OSS is good for point solutions, but integration among projects is often left up to the user, or delivered via add-on software and services from companies such as Gluecode, SpikeSource, and OpenLogic. Yet even these solutions that combine multiple open source packages pale in comparison with the integrated offerings from the superplatform vendors such as BEA Systems, IBM, Microsoft, Oracle, and SAP.

The superplatform vendors are striving to balance the best of both worlds: traditional commercial software and shared source or other community involvement. Many of the vendors are leveraging OSS where applicable but continue to add value through integration and other innovations. In doing so, these vendors have a sustainable and profitable market. Commoditization, consolidation, and standardization will likely result in a market split. Some customers will require the full suite of integrated solutions, and will be best served by the superplatform offerings. A second market, in which software isn't considered as valuable, or in which vendors will not have the resources required to remain competitive beyond the basics, will be served by niche players and open source solutions. Burton Group has seen this already with the commercial J2EE superplatform vendors and the open source alternative JBoss.

Recommendations

OSS is more than just free software. It's a process, a community, a set of licenses, and new business models that are changing the IT industry. OSS is, in many cases, inevitable. While a few companies have implemented anti-OSS policies, OSS is already in their environment, either in embedded devices or appliances, or as an infrastructure component in a proprietary software product.

Burton Group has tracked numerous open source projects and distributions ranging from operating system through development frameworks and applications. For many applications, there are compelling open source alternatives that should be considered. The landscape has changed noticeably over the past few years. Open source is no longer unproven or uncharted territory. Many organizations have been successful implementing Linux, Apache, JBoss, and other open source alternatives.

Cost/Benefit Analysis Is Essential

Don't embrace OSS simply because it's free. Software licenses are less than 5% of the total cost of ownership of most software projects. To consider the true cost and risks of an OSS deployment, look beyond software licenses to migration, training, implementation, support, and long-term maintenance. Will your business perform these tasks internally, depend on Internet OSS resources, or rely on third parties? How important and useful are SLAs and vendor support organizations? Can you transition your existing developers and administrators to an OSS alternative without major

retraining costs? What about legal risks and indemnification strategies?

At the same time, consider the freedom offered by OSS. How does your organization value vendor independence, access to source code, and the ability or opportunity to influence a technology? Do best-of-breed solutions give greater choice and competitive advantage over a particular vendor's technical strategy?

Perform a cost/benefit analysis to answer these questions. Determine the quantitative and qualitative weighting of each of these categories for each project for which OSS is a candidate. Re-evaluate as necessary, especially because most OSS projects evolve rapidly. And don't forget to use this information when negotiating with proprietary software vendors.

Go for Infrastructure, Get with the Community

Most successful OSS projects are at the infrastructure layer—operating system, file, print, web server, Domain Name System (DNS), and Internet mail. Look for infrastructure projects in which the services are commoditized and well-defined through standards and specifications. Consider appliances and embedded devices built on OSS technologies such as NAS devices or intrusion detection systems (IDSs). These solutions mitigate many of the OSS support challenges while delivering a lower cost of ownership.

Take note of the corresponding OSS community, ecosystem, and market opportunity. Is it an active and large community? Does the OSS project appeal to a large and diverse set of users? Is the project critical to large vendors and corporations? How quickly does the OSS community respond to changes within its space? Is the OSS community leading or lagging behind the industry in the adoption of new technologies and standards? Is the project at risk of becoming a closed source?

Answering these questions is one method of determining long-term OSS project viability, since the large, dynamic OSS communities tend to be more successful. And even if you're only a user of an OSS project, familiarity with the development process and resources is useful for understanding the future of the project.

Understand and Mitigate Legal Risks, If Possible

OSS licenses are untested in the world's courts. There are even rumblings in the legal community that the GPL may be unenforceable. By far, the biggest challenge is the lack of indemnification against intellectual property violations. Although other issues, such as derived works and contamination, pose real threats to vendors and customers alike, they're much harder to prove, especially when so little case law and precedent exist today.

Become familiar with the OSS licenses, and pay special attention to the issues surrounding derived works, contamination, and indemnification. An OSS project with a nonstandard license merits careful scrutiny as the risks to the developer or user rise above and beyond those incurred by the common OSS license. Businesses should evaluate an OSS license just as they would a license from any proprietary software vendor.

In general, developers and consumers should seek out OSS projects that are covered under the nonviral OSS licenses, such as the MIT, BSD, or Apache license. These licenses eliminate the risks associated with derived works and contamination. If a nonviral OSS project is unavailable, careful evaluation is necessary before adopting OSS projects with viral licenses. Also, be aware that some OSS projects may include or require other OSS technologies; for example, while the Linux kernel is GPL, Linux distributions contain hundreds of OSS packages released under a variety of viral and nonviral licenses. Note, too, that given planned changes to the GPL, websites and web services built upon GPL software may require some disclosure not required by today's GPL.

Note, however, that no OSS license indemnifies the user from copyright, patent, and intellectual property issues. SCO is testing the waters with its lawsuit against IBM and its threats to users. If these attempts are successful, it's likely that others may follow suit. Ask your vendors if they're willing to stand behind their products that are built on OSS technologies. Some do, such as Novell, HP, and Sun, while others, such as Black Duck Software, offer products and services that help identify or rule out potential intellectual property issues.

Finally, don't forget outsourced projects and system integrators. Businesses will often ask for exclusive rights to works developed under contract. However, if these works are based on viral OSS software, you or your system integrator will be responsible for returning these modifications to the OSS community, which may include your competitors.

OSS provides considerable benefits that can give your business a competitive advantage. However, go into OSS projects with reasonable expectations; cost/benefit analysis is essential, as is an understanding of the business and legal risks that OSS brings.

The Details

What is open source software (OSS)? On the surface, one might answer that it's any software binary that is released with its corresponding source code. Looking deeper, OSS is more than just a new software licensing model. It's also a community, a development process, and a significant force within the software industry that is changing behemoths such as Microsoft, IBM, and Sun Microsystems.

OSS concepts and general licensing requirements have existed for several years. The actual term "open source software" was coined in the late 1990's to describe software that is licensed under specific terms.

The Open Source Initiative (OSI) has defined OSS as software released in accordance with the following principles:

> **1. Free Redistribution**
> The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a

royalty or other fee for such sale.

2. Source Code
The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works
The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code
The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups
The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor
The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License
The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product
The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must be Technology-Neutral
No provision of the license may be predicated on any individual technology or style of interface.[4]

No official organization owns the definition of OSS. Several organizations, such as the Free Software Foundation (FSF) and OSI, help define and defend these open source principles. In addition, both of these organizations will review and provide their stamp of approval for software licenses that adhere to these open source principles.

There are literally tens of thousands of OSS packages. For example, as of May 2005, the OSS repository SourceForge listed almost 100,000 projects with more than one million registered users. Some of the more well-known OSS packages include:

- **Linux,** which has a UNIX-like operating system and related services; although Linux is technically just the operating system kernel, it is commonly used to describe distributions that package the kernel with other components such as filesystems, desktop environments, and management utilities (most Linux distributions include over a thousand open source packages in addition to the Linux kernel)
- **Apache,** which is the Internet's most popular web server
- **MySQL,** which is an open source database
- **Samba,** which is a Common Internet File System (CIFS) implementation for emulating Windows file services, available in Linux distributions and embedded in hardware devices like network attached storage (NAS)
- **JBoss,** which is a Java 2 Platform, Enterprise Edition (J2EE)-compatible Java application server
- **Tomcat,** which is a Java servlet engine
- **OpenOffice,** which is a collection of open source office productivity tools that includes a word processor, spreadsheet, and presentation software

Software Licenses

The OSI principles are the basis of OSS, but they do not constitute a software license. There are dozens of open source licenses, and any individual or corporation is free to create its own license. Most OSS developers use one or more of the popular OSS licenses rather than create their own, unless there's a particular risk that requires a new OSS license. For example, Apple has created the Apple Public Source License, which was submitted to and approved by OSI, but only acknowledged (not fully approved) by the FSF.

As of May 2005, the OSI is tracking and has approved 57 licenses. Although there's no single authoritative source, the OSI has emerged as the de facto group for classifying and approving open source licenses, and carries considerable weight among the open source community and vendors. For example, Sun worked closely with the OSI to gain

approval for the Common Development and Distribution Licenses (CDDLs), which Sun then used as the license for its Solaris 10.

In general, the community prefers that developers use an existing license rather than create a new license. This provides developers a twofold benefit—they can focus on developing code rather than legal licenses, and yet take some comfort in knowing that their software license is generally understood by all.

There are dozens (possibly hundreds) of OSS licenses. However, most OSS software is released under one of the three common licenses, and these licenses are often the basis for new OSS licenses. Among these licenses, the GPL is by far the most popular, but Berkeley Software Distribution (BSD)/MIT/Apache and LGPL use is roughly equal. A quick analysis of SourceForge's software map, broken out by license type, indicates that the GPL is the license of choice among OSI-approved licenses, gaining almost 70% of all software projects, with BSD/MIT/Apache and LGPL tied at around 11%. Note, however, that this is just one OSS project repository (albeit the largest). Other repositories, such as the Apache Software Foundation, may favor other licenses.

GNU General Public License (GPL)

The GNU GPL, commonly called the GPL, is the most commonly used license, with the FSF claiming that more than half of all OSS is released under the GPL. The GPL is the most controversial license because it has clear requirements that prevent any individual or corporation from hijacking or profiting exclusively from the software or any works derived from the software. The GPL permits individuals and companies to profit from GPL-based software, as long as that software (in its entirety) is available at no charge.

The complete GPL and additional information can be found at http://www.fsf.org/licenses/gpl.html.

GNU Lesser General Public License (LGPL)

The GNU LGPL, commonly called the LGPL, is less restrictive than the GPL. The LGPL is often used for libraries (packages of useful software routines), and in fact was previously known as the Library GPL. The primary difference between the GPL and LGPL is that the LGPL allows other programs to use library functions through dynamic linking without restrictions on those programs (derived works). However, the contents of the library itself remain under the GPL. The LGPL allows software developers to call LGPL software from their applications but does not require that those applications be released under the GPL or LGPL.

The complete LGPL and additional information can be found at http://www.fsf.org/licenses/lgpl.html.

The BSD, MIT, and Apache Software Licenses

The BSD license, and the popular MIT and Apache derivatives, are the most liberal of the common OSS licenses. In general, these licenses grant all rights to use the software while retaining the copyrights, with some restrictions on copyright acknowledgement depending on the license. Unlike software licensed under the GPL, an individual or organization may take BSD-licensed material as the basis of a closed, proprietary software product without any obligation to release the source or bindery back to the

open source community. In fact, both Apple and Microsoft have used BSD-licensed software in their proprietary software products.

There are many, many other OSS licenses. The OSI maintains a list of approved open source licenses (http://www.opensource.org/licenses/), and the FSF maintains a list of GPL and non-GPL compatible free software licenses (http://www.fsf.org/licenses/license-list.html).

Other OSS Licenses

Why are there so many open source licenses? There are several answers. First, a number of corporations, such as Apple, Netscape, and others, have tailored their OSS licenses for additional control or risk aversion. Second, the various licenses give developers greater choice in how their software will be utilized. For example, developers may choose the GPL if they want others to benefit but not profit from their intellectual property. Or developers may choose a BSD-style license if they desire adoption of their intellectual property without restriction—for example, Intel uses a BSD-derived license so that others may build proprietary products leveraging Intel's work (thus increasing the market for more Intel hardware and software).

Dual Licensing

It's also important to note that some open source projects are dual licensed, meaning that they are released under multiple software licenses. For example, the open source database software MySQL is released under both the GPL and a proprietary license. MySQL requires a proprietary license for those MySQL-based products that are not released under the GPL or another OSI-approved license. In return, MySQL provides some indemnification and advanced features that are not available under the GPL of MySQL.

Microsoft Shared Source License Programs

Microsoft's Shared Source License programs are not an OSS license, but they do provide some advantages from the OSS licensing model. Microsoft, like any other proprietary software developer, is at considerable risk because of OSS. The company recognizes that open source provides some qualities desired by its proprietary software customers. In response to the OSS, Microsoft has created several software licensing programs, known as Shared Source, that grant limited access to certain Microsoft products.

Microsoft provides Shared Source License programs based on customer type (enterprise, system integrator, original equipment manufacturer [OEM], government, and academic) and by product (Windows CE, .NET Passport, and some sample code). These programs enable customers to study, inspect, and debug selected Microsoft products. While the Shared Source program does provide greater insight into the internal workings of selected Microsoft products, it's not nearly equivalent to the rights and flexibility associated with OSS.

For more information on Microsoft's Shared Source License program, see http://www.microsoft.com/resources/sharedsource/default.mspx.

License Enforcement

Although OSS is sometimes mistakenly referred to as "public domain," there's an important legal distinction between the two terms. Public domain software has no copyright holder—people are free to do whatever they wish with it. OSS, on the other hand, is copyrighted work released under a liberal yet specific software license. As such, OSS software licenses can be enforced.

It's not uncommon to find proprietary software vendors such as Microsoft, Oracle, Sun, and others taking legal action to enforce their software licenses. However, in the open source community, it's difficult, if not impossible, for an individual software developer to enforce a software license, especially if the violation is by a larger corporation. Many OSS releases are the work of just a few individuals who have neither the time nor the resources to track down and take legal measures against license violators. Even so, the open source community has found a relatively successful strategy for license violations: peer pressure.

An OSS license violation is publicized throughout the open source community, and the community responds, often on principle rather than personal affiliation with the software in question. The OSS community can place pressure on a corporation to comply with the OSS, often by badgering through e-mail or web postings, and by threatening product boycotts. In general, most violations are unintentional and are quickly resolved, with the offending party either removing the code in question, or complying with the OSS license and releasing their modifications back to the open source community. TiVo, Dell, Linksys, and many others have unintentionally violated the GPL but responded quickly to the OSS community by making the source code publicly available. nVidia mistakenly incorporated GPL source code into their XFree86 driver, but quickly removed the GPL software after being notified of the violation.

Most OSS license violations involve GPL software primarily because other licenses, such as the BSD/MIT/Apache, permit the inclusion of software source code into proprietary products (with restrictions). The FSF maintains and is a staunch enforcer of GPL and LGPL, and, unlike independent developers, has the in-house legal expertise to enforce the GPL.

However, only the copyright holder is legally authorized to act upon license violations. If the developer retains the copyright on the work, the FSF will offer assistance and guidance for license enforcement. At the same time, the FSF encourages developers to consider assigning the FSF the copyright to their work, so that the FSF is authorized to act upon license violations. For more information, see http://www.fsf.org/licenses/why-assign.html.

OSS Development Principles and Processes

The OSS development process is quite a bit different from that for traditional proprietary software. Although there is no single OSS development process, most projects tend to use the same development principles and process.

Eric S. Raymond's "The Cathedral and the Bazaar" is an oft-cited paper that describes the fundamental principles of OSS development. Raymond examines the goals and methodologies of the traditional, highly structured software development process (the

cathedral) with the more loosely structured, open source development process (the bazaar).

The formalized cathedral development model begins with product managers collecting and prioritizing product requirements based on customer input and company goals. Engineering managers then analyze the product requirements and commit to timelines based on engineering resources. Developers, who may or may not have a personal interest or experience in the product under development, toil for weeks or months before releasing an end product. The product is initially beta tested by a limited group before it is finally publicly released. After the final release, the process is repeated, typically on an 18–36 month development cycle.

Raymond describes the bazaar development model of a new product by stating that "Every good work of software starts by scratching a developer's personal itch."[5] Additional developers may be recruited to the project or may find the project because they, too, are scratching a similar itch. Instead of a formal road map and feature set, features are added on an as-needed basis, often driven by real-world problems that require an immediate solution. Any developer is welcome to download the latest source code from the project repository and build a customized solution.

Neither software model is necessarily correct; both should (and do) result in usable software products. These development models are sometimes viewed as opposite ends of a development spectrum. In practice, real-world software development, whether proprietary or open source, falls somewhere between the pure cathedral and pure bazaar models. For example, many proprietary software companies are borrowing ideas from the OSS development process to foster collaboration and speed up the development process. VA Linux's SourceForge software, and CollabNet's SourceCast are attracting an increasing number of proprietary software developers who wish to implement OSS-style development in a proprietary software project. Likewise, some OSS development processes are used by proprietary-style product management to organize and deliver features.

Both development methods have benefits, but the OSS development process provides greater flexibility for developers and users alike. For example, a proprietary software company is driven not just by customer requirements, but by company goals. Such a company may intentionally limit backward compatibility or interoperability to force a customer upgrade or to create an upsell opportunity. OSS, on the other hand, allows anyone to download and build a customized version (thus ensuring compatibility or interoperability) but at the expense of additional developer and testing resources.

OSS software development is commonly platform-independent—although different versions may exist for different hardware architectures or operating systems, there's no revenue issue to drive platform affinity. As a result, many of the development tools, methodologies, and source code control systems are, themselves, platform independent. Although this may increase the complexity for unfamiliar or rookie developers, it provides the greater flexibility that supports a wider developer and user audience.

Sun's New Java Licenses

In March 2005, Sun announced a new licensing scheme for Java. It's part of a larger plan, referred to as "Project Peabody," which is designed to increase transparency into the development process of the Java specifications and reference implementations. Project Peabody has the following goals:

- Simplify Java licensing practices
- Increase transparency of the Java development process
- Give developers a chance to contribute
- Maintain Java compatibility

Sun has devised three new licenses to replace the Sun Community Source License (SCSL), which is the extremely complex license Sun now uses for Java source code. These licenses will apply to Java 2 Platform, Standard Edition (J2SE) 6.0 (Mustang). The three new licenses include:

- Java Distribution License (JDL)
- Java Research License (JRL)
- Java Internal Use License (JIUL)

JDL applies to anyone who wants to distribute a J2SE implementation for any purpose other than research. The license requires that the J2SE implementation must prove J2SE compatibility by passing the Test Compatibility Kit (TCK).

JRL is designed to support the needs of the research community. It permits research groups to develop and share J2SE implementations for research purposes. To redistribute a derivation of J2SE for any purpose other than research, one must obtain a JDL and pass the TCK. J2SE 5.0 ("Tiger") is now available under JRL. Sun also plans to release weekly builds of Mustang under the JRL.

JIUL (pronounced "jewel") is designed to support the needs of the average company that uses Java. It permits "users" (personal, academic, or corporate entities) to make fixes and enhancements to the Java source code for their own internal use. Such users are not required to pass the TCK, but they also may not share their fixes with other users (except for research purposes). JIUL relies on the honor system to ensure compatibility, but what that really means is that compatibility is not ensured if you make modifications.

Sun has also developed a new set of contribution agreements. Developers are encouraged, but not required, to contribute bug fixes and modifications back to Sun. Contributing developers retain rights to their code, but they must also grant all rights to the code to Sun.

Additional OSS Resources

There are thousands of articles and hundreds of sites dedicated to the evangelism and developer support of OSS software. Here are several of the more popular and useful sites:

- www.fsf.org**:** The Free Software Foundation (FSF) provides background about free software and OSS history, ideology, and licenses. The FSF maintains the

GPL and LGPL and is one of the strongest legal enforcers of these licenses.

- www.opensource.org**:** Like the FSF, the Open Source Initiative (OSI) evaluates OSS licenses and provides its stamp of approval for licenses and OSS works. OSI is a bit less radical than the FSF in their support of OSS. For example, whereas the FSF discourages anything but GPL-covered works, the OSI recognizes and supports non-GPL open source licenses.
- www.freestandards.org**:** The Free Standards Group (FSG) is a nonprofit organization dedicated to promoting open source software and standards. Among other activities, the FSG promotes the Linux Standard Base (LSB), a project to provide application programming interface (API) and filesystem consistency across Linux distributions, and OpenI18N (I18N is an abbreviation for Internationalization) a project that addresses localization and internationalization of Linux distributions and applications.
- www.sourceforge.net**:** SourceForge is a large collection of OSS developers (more than 680,000 registered users) and OSS projects (more than 67,000). SourceForge not only hosts the developer webpages and resources for OSS projects, but also is a portal for news and activity in the OSS community.

Conclusion

Open source software (OSS) brings new opportunities and risks to information technology (IT) vendors and customers. Software commoditization is being profoundly affected from the infrastructure layer and moving up to the application layer. As a result, OSS is changing the strategies and business models of hardware and software vendors and system integrators. Customers should understand the risks and rewards of OSS, and should formulate strategies that bring OSS benefits to their IT departments while mitigating the business and legal risks.

Notes

1 Free Software Foundation. “Frequently Asked Questions about the GNU GPL.” *FSF.org*. http://www.fsf.org/licensing/licenses/gpl-faq.html.

2 “Shared Source Initiative Frequently Asked Questions.” *Microsoft.com*. 1 Mar 2005. http://www.microsoft.com/resources/sharedsource/Initiative/FAQ.mspx.

3 “Red Hat, Inc. Form 10-K/A: 2004 Annual Report.” 9 Aug 2004. http://media.corporate-ir.net/media_files/IROL/67/67156/reports/rhat_093004.pdf.

4 Bruce Perens and the Open Source Initiative. “The Open Source Definition.” *OpenSource.org*. Jun 1997. http://www.opensource.org/docs/definition_plain.php.

5 Eric S. Raymond. “The Cathedral and the Bazaar.” Aug 2002. http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s02.html.

Author Bio

Gary Hein

Vice President Service Operations

Emphases: Directory and Web services, open source software, Microsoft .NET and J2EE application development platforms

Background : 14 years of experience in the networking industry, including the design and deployment of directory services and directory-enabled solutions for Fortune 500 customers. Additional responsibilities: technical sales and support, consulting, marketing, keynote presentations, and the teaching of educational courses (including the development of associated manuals and training guidelines).

Primary Distinctions: Pioneered Novell's move towards open source software with the launch of forge.novell.com in 2003. Recipient of 1998 Novell President's Award. Published numerous high-level business-oriented and technical white papers analyzing Novell's NDS and Microsoft's Active Directory. Co-authored "Novell's Guide to Integrating NetWare 5 and NT" (IDG, 1999). Designed and built Novell's first billion-user NDS tree as a proof of Novell's NDS v8 technology.