

THE LEADING EDGE FORUM PRESENTS:

# Open Source: Open for Business



EXPERIENCE. RESULTS.

## ABOUT THE LEF DIRECTORS



### PAUL GUSTAFSON

Director, Leading Edge Forum, and Senior Partner,  
CSC Consulting Group

Paul Gustafson is an accomplished technologist and proven leader in emerging technologies, applied research and strategy. As director of the Leading Edge Forum, Paul brings vision and leadership to a portfolio of programs that make up the LEF and directs the technology research agenda. Astute at recognizing technology trends, how they inter-relate, and their implications for business, Paul brings his insights to bear on client strategy, CSC research, leadership development and innovation strategy. He has published numerous papers and articles on strategic technology issues and speaks to executive audiences frequently on these topics.

[pgustafs@csc.com](mailto:pgustafs@csc.com)

CSC's **Leading Edge Forum** is a global thought leadership program that examines the technology trends and issues affecting us today and those that will impact us in the future. As part of the CSC Office of Innovation, the LEF explores emerging technologies through sponsored innovation and grants programs, applied research, awards for the most innovative client solutions, and alliances with research labs. The LEF examines technology marketplace trends and best practices, and stimulates innovation and collaboration among CSC, our clients and our alliance partners.

In this ongoing series of reports about technology directions, the LEF looks at the role of innovation in the marketplace both now and in the years to come. By studying technology's current realities and anticipating its future shape, these reports provide organizations with the necessary balance between tactical decision making and strategic planning.

### WILLIAM KOFF

Vice President, Leading Edge Forum

Bill Koff is a leader in CSC's technology community. He chairs the Leading Edge Forum executive committee, whose members are the chief technologists from each of CSC's business units. Bill plays a key role in guiding CSC research, innovation, technology thought leadership and alliance partner activities, and in certifying CSC's Centers of Excellence. He advises CSC and its clients on critical information technology trends, technology innovation and strategic investments in leading edge technology. A frequent speaker on technology, architecture and management issues, Bill's particular areas of interest are system architecture, digital disruptions and the open source movement.

[wkoff@csc.com](mailto:wkoff@csc.com)

# Open Source: Open for Business

## CONTENTS

### 2 GETTING DOWN TO BUSINESS

#### 6 TREASURE CHEST: TECHNOLOGY TRENDS

6	Culture of Community: <i>The heart and soul of open source is community.</i>	49	At Your Service: <i>Service opens up new business opportunities.</i>
11	Moving Up the Stack: <i>Open source is not just Linux.</i>	55	Invisible Man: <i>Open source is all around us.</i>
23	Mission Critical: <i>Open source is industrial strength.</i>	60	Market Force: <i>Open source increases competition, challenging established market powers.</i>
31	Sweet Spot: <i>Open source yields targeted savings.</i>	65	New Domains: <i>Open source lives in many domains.</i>
44	Software Revolution: <i>Open source accelerates development and incubates new ideas.</i>	69	Fun Factor: <i>Open source gets your creative juices flowing.</i>

### 74 LEGAL AND BUSINESS ISSUES

### 84 GETTING STARTED

### 87 APPENDIX: HANDY WEB SITES

### 91 ACKNOWLEDGMENTS

## GETTING DOWN TO BUSINESS

Something disruptive is happening when:

- organizations operate on the premise of paying \$0 for new software infrastructure, demanding justification for any purchase costs above that
- a global software development community over 800,000 strong challenges the leading software vendors like no competitor can
- organizations achieve time-to-market, innovation and product quality like never before
- commodity computing platforms bring significant price-performance benefits to more and more organizations, defying proprietary approaches
- organizations eye the methodology of the global development community to improve their own way of developing software
- governments around the world issue directives steering away from proprietary software
- software vendors are forced to prove their case.

That disruption is open source, the software development model made popular by the Linux operating system. With Linux as the star, there is a rich cast of open source software available today for Web servers, application servers, databases, content management, office systems, browsers, development tools, security and more. Open source brings about the reorganization of millions of software developers into global collaborative communities, amassing a strength orders of magnitude greater than what is possible in the proprietary software realm.

Organizations of all kinds are consciously adopting open source software for critical business needs: Deutsche Börse Group, Deutsche Bank, the Danish government, BlueScope Steel, NASA, the Associated Press, JPMorgan Chase and Google, to name a few. There have been many government initiatives around open source software, as governments in Brazil, China, India, Korea, Japan, Europe,

Australia and the United States, as well as the United Nations, consider open source policy and options. And large information technology vendors such as IBM, Intel, Hewlett-Packard, Oracle, SAP, Sun Microsystems and Dell are supporting open source.

The lure of open source software is that it is “free” in the sense that anyone can use it, modify it, create derived works from it, and redistribute it – and there are no license fees. You have access to a worldwide development community that improves, adapts and fixes the software, often much faster than in the proprietary vendor world. You are not beholden to a vendor for fixes and enhancements; there is no vendor product lock-in.

At the same time, open source software is not a silver bullet; it is not inherently good just because it is open source. Open source software is not appropriate for every situation; it will not displace proprietary software overnight. There is plenty of good proprietary software on the market, which can and should be deployed. Interestingly, though, the lines are blurring between proprietary and open source as software vendors begin recognizing and applying open source software in their products and services.

In the end, organizations need to focus on the software that best fits the business case. That is why open source needs to be on the business radar screen. Open source has become a viable IT option that is growing stronger by the day. Issuing a wake-up call, *CIO Magazine* has asserted, “CIOs who don’t come to terms with this [open source] revolution in 2003 will be paying too much for IT in 2004.”<sup>1</sup> Gartner has stated that companies will be considering open source options for 30 percent of new systems through 2004.<sup>2</sup>

Open source is open for business.

## UNLEASHING IDEAS AND INNOVATION

Open access and collaboration are at the core of the open source movement. If information and ideas “want to be free,” the Internet is

...the Internet is the key enabler propelling the open source movement and the free flow of ideas in the 21st century.



```
applicati
}
fireContainer

blic void addCh

Wrapper oldJsp

if (!child in
throw new
(sm.g

)

Wrapper wrapper
boolean isJsp

if (isJspServ
oldJspServ
if (oldJsp
remove

}

curity.declarePr
ef* *cook*(self,
*""
Pre-parse this
""
cooklock, acqui
try:
self._v_bl
self._v_co
finally:
cooklock.r

ef* *evaluatePre

*return* DTMLC

ef* *renderMidse
```

the key enabler propelling the open source movement and the free flow of ideas in the 21st century. Open source is a philosophy of idea generation and development that challenges many business basics: how we do business (R&D that is not proprietary), the fundamentals of intellectual property (copyrights that are less restrictive), and the nature of software development (from closed to open, from proprietary to participatory).

Open source is a movement that is technical, political and sociological.

The open and collaborative nature of open source fuels innovation. In the *Harvard Business Review* article “Breaking Out of the Innovation Box,” John Wolpert of IBM contends that innovation is spurred by an open, rather than insular, process: “Initiatives must gain access to and leverage from the insights, capabilities, and support of other companies without compromising legitimate corporate secrets.”<sup>3</sup> By tapping a worldwide development community that knows no

corporate boundaries, and sharing that expertise, the open source movement breeds innovation, be it for corrections, enhancements to existing functionality, or new functionality.

Vendor repercussions are being felt, from the SCO Group’s lawsuits alleging Linux infringes on SCO’s intellectual property rights to the Microsoft-Sun accord that some say can be partially linked to open source and the need to present a united vendor front. In his book *The Business and Economics of Linux and Open Source*, Martin Fink likens the impact of open source on the software industry to the impact of generic drugs on the pharmaceuticals industry. The emphasis is on commoditization and lower prices, resulting in wider access for consumers and increased competition for vendors.

Indeed, open source places the scarce resource of software into everybody’s hands, the way the Gutenberg press placed the scarce resource of texts into everybody’s hands. The open, collaborative approach levels the playing field, enabling anyone to contribute and defying the big hand of the corporation. Open source is a movement that is technical, political and sociological.

*Open Source: Open for Business* identifies 10 trends that are defining the open source movement. The report probes technology, mission-critical applications, savings opportunities, and legal and business issues, concluding with core propositions for getting started on your organization's open source journey. In the conclusion, the trends are put in perspective with a matrix rating each trend against a set of key business drivers. (See chart page 84.)

These key business drivers underlie the trends and are putting open source software at the center of business strategy. Open source is decreasing time to market for key products and services (e.g., Deutsche Bank), offering new possibilities for solving critical business problems (e.g., BlueScope Steel), and providing business interoperability through standardization and technology transparency (e.g., Danish Ministry of Finance).

Read on to understand more about open source – the movement, the technology, the business implications.

```
class StandardContext
    implements ContainerBase,
        ServletContext, Serializable

    private transient Log log = LoggerFactory.getLog(StandardContext.class);

    public StandardContext() {
        super();
        pipeline.setBasic(new StandardContextValve());
        namingResources.setContainer(this);
        broadcaster = new NotificationBroadcasterSupport();
    }

    public void addApplicationParameter(ApplicationParameter parameter) {
        synchronized (applicationParameters) {
            String newName = parameter.getName();
            for (int i = 0; i < applicationParameters.length; i++) {
                if (newName.equals(applicationParameters[i].getName()) &&
                    !applicationParameters[i].getOverride()) {
                    return;
                }
            }
            applicationParameters[i].setOverride(true);
        }
    }

    // ... other methods ...
}
```

# TREASURE CHEST: TECHNOLOGY TRENDS

The open source movement is not confined to a limited group of products or people but is rich in breadth and depth; it is a treasure chest teeming with technologies and best-practice methods. This section explores 10 trends core to the open source movement:

- Culture of Community
- Moving Up the Stack
- Mission Critical
- Sweet Spot
- Software Revolution
- At Your Service
- Invisible Man
- Market Force
- New Domains
- Fun Factor

## CULTURE OF COMMUNITY:

The Heart and Soul of Open Source Is Community

The heart and soul of the open source movement is community. The community brainstorms and develops the software, fixes it, enhances it. The community reuses it to spawn new innovations. The community shares and feeds off expertise in a loosely-structured meritocracy. The culture is about participation, not profits.

What motivates people, then, to get involved in open source? Why do they volunteer their time? What do they learn that can be passed back to their companies as best practices?

### CULTURE OF COMMUNITY

Community in IT has been around since the dawn of computers, the 1940s, when software was created for mainframes and freely shared among the limited

few at universities and government-funded labs who knew how to use computers – an elite corps of programmer enthusiasts. Today the network effect<sup>4</sup> of the Internet enables programmers around the world, and users, to participate in communities more deeply.

Why should people care about community? Naysayers will argue that people can only be motivated by money. But that's not true: look at the myriad volunteer organizations around the world, from the Red Cross to the Peace Corps. Much of the world lives by volunteer efforts benefiting the greater good; why should the IT sector be any different? Open source has a different social heart beat, to be sure, but today's IT sector need not be resistant to the notion of community.



Indeed, despite the presence of a strong proprietary software industry, software development practices historically have been rooted in community and sharing. The Internet itself was an open, collaborative effort, starting with the linking of four nodes in 1969. This was followed by many initiatives that became precursors to the modern open source movement, including Richard Stallman's GNU project, MIT's Project Athena and public domain software. (Stallman founded the Free Software Foundation and authored the General Public License, or GPL, the most widely used open source license today. See Legal and Business Issues.)

The open movement gathered steam in the early 1990s with Linux, and the commercialization of the Internet in the same decade turbocharged the effort. In 2000, Eric Raymond famously characterized this new form of development in his "cathedral and bazaar" paper.<sup>5</sup> Proprietary development was like a cathedral: massive, closed, slow and even reverent. Open source development was like a bazaar: flexible, open to new ideas and approaches, faster and very independent.

At the heart of this development is the community. Typically, the community is seeded by a single person, a genius programmer who creates the initial code: Stallman for GNU, Linus Torvalds for Linux. The community comes in to enhance what the initial programmer has started.

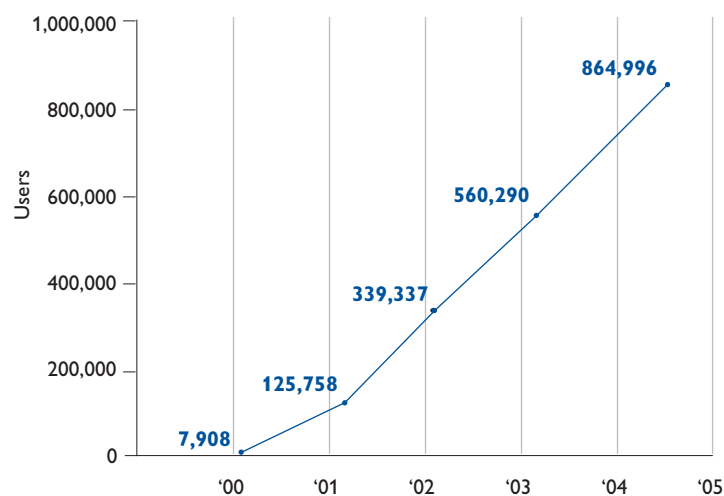
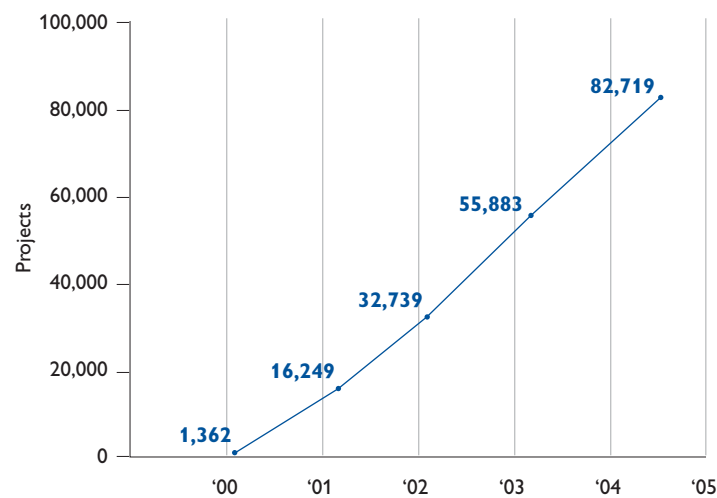
As Raymond wrote in describing Linux: "From nearly the beginning, it was rather casually hacked on by huge numbers of volunteers coordinating only through the Internet."<sup>6</sup> This community approach was the jewel, Raymond asserted: "The most important feature of Linux...was not technical but sociological."<sup>7</sup>

## THE COMMUNITY TAKES OFF

By the late 1990s the network effect was in full swing, as was the momentum around Linux development, the most prominent open source project to date. The open source development community had become a credible force, and it continues to grow strong today.

One sign of community credibility is SourceForge, the world's largest open source software development Web site. With over 82,000 hosted projects and more than 860,000 registered users, SourceForge is a community of communities for developers. SourceForge lends

## THE OPEN SOURCE COMMUNITY TAKES OFF



Soaring growth in the number of projects and registered users at SourceForge, the primary site for open source software development, shows how the open source movement has accelerated in the last few years.

Source: Sourceforge

credibility to the communities, demonstrating that they are worthy of being managed with a sophisticated site that provides project hosting, version control, bug and issue tracking, mailing lists, e-mail archives, project management and collaboration resources. This is not a renegade process but a well-managed one.

Who are these open source developers? A survey by Boston Consulting Group in 2002 of developers using SourceForge found that respondents were, on average, 30 years old and had 11 years of programming experience. These are not just college students or people in their first jobs but experienced IT professionals. Many held paying jobs as programmers (45 percent), system administrators (6 percent) or IT managers (6 percent). Still, one cannot overlook the student ranks (20 percent) and academia (7 percent).<sup>8</sup>

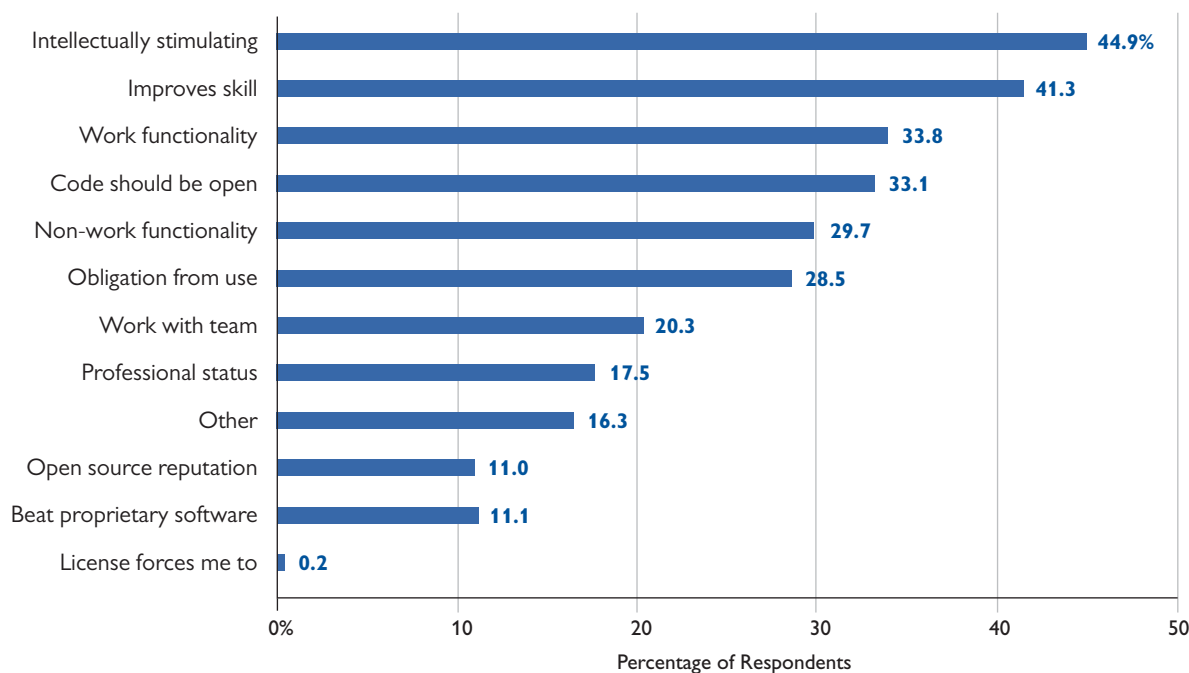
Another measure of developer credibility comes from GForge, an open source collaborative development envi-

ronment. At GForge.org you can search for developers with specific skills, the way a corporation can query a skills database for capabilities. Developers are ranked by each other in a sophisticated manner: the higher your own ranking, the more weight given to your rankings of others.

In addition to people, community credibility stems from the product – its functionality, performance, technical elegance and level of use. “Apache software is so good it drives you to join its open source community,” observes Gabor Herr, a solution architect for CSC’s e-Business and Technology Center in Wiesbaden, Germany. Conversely, a poor or esoteric product will be less likely to draw people to its community.

Community credibility is an underlying motivator for joining an open source project. The lure of open source includes solving technical challenges; drawing on the best minds around the world; the prospect

## DEVELOPER MOTIVATIONS



Note: Question asked for top three motivators of free/open source software participation, n=684.

Source: The Boston Consulting Group, in cooperation with the Open Source Developer Network, surveyed developers participating in software projects on SourceForge.net (2002).

of making a contribution the rest of the community can use; the enhanced skills and reputation (marketability) that comes from being an active member of the community; and the potential for providing fee-based services for open source software (see At Your Service). As well, developers are motivated by the opportunity to branch out and work with products they don't normally work with in their day jobs – say, video programming – and they are also motivated by pure fun. (See Fun Factor.)

### SHARE THE WEALTH

Best practices learned from the open source community can be passed back to the corporation and adopted in its methodologies. Developers involved with open source, either on their own or as part of a corporate project, can “share the wealth” of open source ways, opening up a new world of development and project management in the corporation.

This is happening at CSC, where a small team's initial foray into an open source project has led to standard development practices that incorporate open source methodology and mentality.

“We have learned to use approaches and behavior similar to what we found in the open source community,” explains Stefan Höhn, a lead architect for CSC's e-Business and Technology Center in Wiesbaden, Germany. “From working on several open source projects, we gradually adopted many best-of-breed open source processes into our development methodology. We became so competitive that we were able to outperform off-shore projects in terms of development speed, quality and price.”

In 2001, CSC began working on a system for a subsidiary of Deutsche Börse Group (German Stock Exchange) in Frankfurt am Main. CSC was asked to reduce overall costs by using open source software for parts of the system. The CSC team chose some well-established open source components for the underlying infrastructure. Although this was not CSC's first contact with open source software, it was the first time the team had worked on a project that relied solely on an open source community for support.

This meant the team had to learn to get support without calling a vendor hotline. Team members combed the Internet for information, e-mailed people they had no business relationship with, asked questions in online forums they didn't know existed, and dug into documentation that ranged from poor to fantastic.

Surprisingly, it worked, and led to a significantly larger project. (See Mission Critical.)

“It turned out better than we expected,” Höhn recalls. “However, we had to learn to cope with problems in a totally different way than we were used to.”

### LESSONS FROM THE COMMUNITY

Four approaches emerged from the open source world that infiltrated CSC methodologies: collaborative source code management, collaborative development, automated regression testing, and a more agile development methodology.

“During the team's first contact with the open source world, we discovered the social dynamics of the development communities and realized there was a lot to learn from their processes,” Höhn says. “So we adopted our first collaboration tool, for source code management.”

This was CVS, the code version management system used by SourceForge. Today CVS is also used in CSC's GPES Unix Software Factory, housed in the U.K., to help manage internal Unix systems.

The success of the initial open source project compelled Deutsche Börse Group to go forward with open source on a larger scale, for a reporting system to support its regulations enforcement process. The customer mandated that the system be developed without incurring any software licensing fees (beyond the proprietary database already being used). This meant open source.

The project team was distributed, located at CSC offices, the client site and a subcontractor's office. The lack of face-to-face contact among the developers, coupled with CSC's experience with open source development, led to the creation of the CSC Toolbox, a Web-accessible collaborative environment for the

```
int=None, REQUEST=None):
    rendered():
    red(self.pageType
    e().render(self
    ear_cache=0):
    f.clearCache()
    e(*'prerender
    self.pageType(
    t(Permissions.Vt
    REQUEST=None):
    v cached render
    v_*):
    v_cooked '*):
    v_cooked '*):
    v_blocks '*):
```

project team. As a first step, the team incorporated CVS into the Toolbox to allow distributed access to the source code.

...employees can participate in open source projects and communities and bring back best-of-breed approaches to the corporation at large.

However, whereas in SourceForge all information is freely available, CSC's project required more security. The team added encryption and authentication to the Toolbox, to ensure that the sources were not compromised during transmission and that only authorized personnel had access to the collaborative environment.

As the project proceeded, more services were added. One was an issue-tracking tool (Mantis), which enabled the client to participate in the collaborative environment.

Testing was another area where open source concepts were adopted. A distributed development environment can lead to loss of control with software versions. In the open source world, this is addressed with unit and regression tests, which the CSC team adopted. Besides a noticeable quality improvement, the approach led to faster development and an earlier project completion date.

Later, when the team was asked by a different customer to review its development process for a large, strategic project, the team recommended a more agile methodology akin to open source methods. The customer's conventional development model was plagued by labored workflows and inadequate quality.

Based on the CSC team's proven development model, the customer changed its model to be more quality driven (rather than time driven) according to CSC's recommendations, and made its quality measures more apparent to management. Test results were published for all to see, from developers to senior managers; everyone could monitor and manage progress.

Naturally, the CSC team used open source software to support the customer's new development environment. JUnit and CruiseControl were the technological cornerstones. JUnit is for writing tests for small units of code, such as an object. CruiseControl builds and runs tests and creates a management dashboard. This is key in a distributed development environment, where testing needs to be fully automated. For this project, a team of 70 developers wrote approximately 1,600 unit tests in just a few weeks.

With this experience, CSC further refined its open source-influenced methodology, which would be used again and again in new projects.

One of these was a cross release management system, a truly agile development effort in which CSC outperformed off-shore competitors. The system was an application created for Deutsche Bank. For the same price as off-shore competitors, CSC provided newer technology, better quality, faster development and reduced risk from having developers on site.

This led to the latest CSC project for Deutsche Bank, to develop one of its mission-critical systems based completely on open source software and using agile development. The agile, open source approach suits perfectly the rapid project schedule, which has deliverables every month.

CSC continues to learn from its open source experiences. As at CSC, employees can participate in open source projects and communities and bring back best-of-breed approaches to the corporation at large. These approaches depend inherently on the community. The community offers a powerful approach for tackling problems, especially complex ones, and maximizing performance. Among other things, open source communities are being used today to translate open source software and applications into rare languages to help bring computing to developing nations. This is the sort of complexity – advancing the globalization of computing through localization – that suits the open source community well.

## MOVING UP THE STACK: Open Source Is Not Just Linux

The development community has been hard at work, for today open source software pervades the software stack. Open source extends from the lowest reaches of Linux at the operating system level to databases, application servers, development tools and, more slowly, business and desktop applications. Open source is not just Linux but a broad collection of technologies that provides a range of functionality and opportunity.

To discover what's in the open source collection, we will examine the software stack using a taxonomy from Flashmap Systems.<sup>9</sup> As we move through the stack using this taxonomy – Enterprise Infrastructure, Business Applications, Commercial Off-the-Shelf Applications – we will see a rich portfolio of open source software for the infrastructure. Higher-level business applications are rare but evolving as open source software gradually makes its way into common business processes.

A word of caution: It is debatable where the software stack actually starts and ends in any one organization given today's trends toward outsourcing and blurred lines between businesses. We are using a subset of Flashmap's taxonomy to highlight key portions of the stack common in most organizations and relevant to uncovering the best in the open source collection. What we discover is only a small portion of all the open source software that exists.

### ENTERPRISE INFRASTRUCTURE

*The physical hardware used to interconnect computers and users, and the software that supports the flow and processing of information.* (See chart page 12. Our focus is software.)

#### Platforms

*The execution foundation for all systems and applications software.*

**Linux**, the most well-known software in the modern open source movement, is the number two server operating system behind Microsoft's Windows NT.

IDC predicts that Linux platform revenue will increase at more than four times the overall industry average for all platforms through 2007.

Besides Linux, there is a host of other open source operating systems, all Unix derivatives like Linux. **BSD Unix** (Berkeley Software Distribution) was one of the first operating systems to become open source. Several open source operating systems have been developed from BSD Unix, including **FreeBSD**, **OpenBSD** and **NetBSD**, an advanced operating system for research and production environments. Additional platforms based on BSD Unix are in various stages of development. In the proprietary world, Sun has announced it is releasing **Solaris**, its flagship version of Unix, to the open source community.

Even specialized operating systems exist as open source. One of these is the **Robotic Operating System**, a behavior-based operating system that runs on top of Unix-like systems.

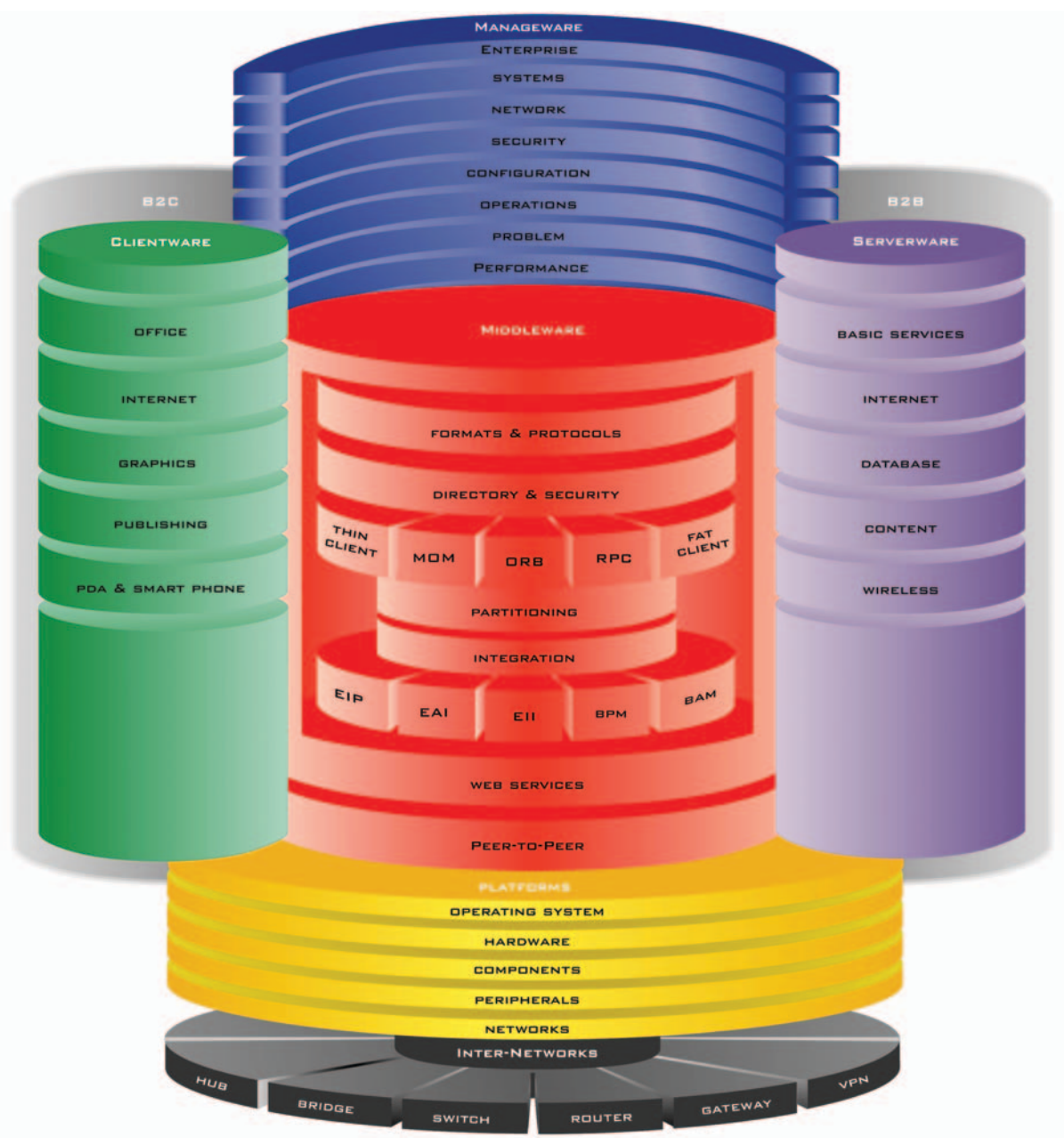
Another important dimension of the platform is how it can be scaled by leveraging the network. Grid computing is one example: a large virtual computer is assembled by linking tens, hundreds or thousands of heterogeneous systems.

One leading open source implementation for grid computing comes from the Globus initiative; its Globus Toolkit is a fundamental enabling technology and considered the de facto standard for grid computing. The latest version of the toolkit, **Globus Toolkit Version 3** (GT3), is an open source implementation. The toolkit is based on the Open Grid Services Architecture (OGSA) and is core to grid computing offerings by major vendors including IBM and Platform Computing.

Another popular implementation is designed around localized clusters and enterprise grid needs and comes from Sun's **Grid Engine** project, an open source collaborative development effort. With the acquisition of



## ENTERPRISE INFRASTRUCTURE TAXONOMY



Copyright © 1993-2004 Jeff Tash, Flashmap Systems, Inc. (www.flashmapsystems.com). All Rights Reserved.

Gridware, Sun has been able to facilitate the deployment of compute farms, the basic building blocks of grid computing.

Elsewhere in the cluster arena, Linux is having a dramatic impact on high performance computing (HPC) clusters; its low cost is reducing previous barriers to entry, and more open source HPC systems

are emerging. For example, the company Linux Networkx has been selected to build the two most powerful Linux clusters in the world, for Lawrence Livermore National Laboratory and Los Alamos National Laboratory. Powering the clusters is [LinuxBIOS](#), an open source BIOS alternative from Linux Networkx that enables low-overhead, highly-scalable clusters of thousands of nodes.

In addition to labs, open source HPC clusters are being put to work in industry. One large energy company is using such a cluster to power its seismic research for oil exploration. (See Sweet Spot.)

Other important open source HPC software initiatives include [Scyld Beowulf](#), [OSCAR](#) and [openMosix](#).

## Middleware

*The “magic glue” that connects clients to servers.*

The Internet and its various protocols have their seeds in government and university research projects, which were the roots of the open source movement. So the first and thus reference implementation of nearly all Internet protocols, including the fundamental [TCP/IP](#), was done as open source software and became part of BSD Unix.

Driven by U.S. export restrictions on cryptographic software, many Internet security tools were developed outside the United States in the 90s. These open solutions fueled innovation by enabling people to bypass the U.S. restrictions. As the tools became commodities, the restrictions were relaxed.

For example, [OpenSSL](#) (formerly known as SSLeay) was written by Eric Young and Tim Hudson, two Australians who started SSLeay to provide a free version of Netscape’s SSL/TLS protocol. Since it was implemented outside the United States, OpenSSL could be used as a full-strength encryption protocol for Internet transmissions (e.g., HTTPS). Today OpenSSL is the cryptography library for many open source products, like the popular Apache (for HTTPS transport), and is widely used inside and outside the United States.

Another important security protocol is SSH, which spawned [OpenSSH](#), an enhanced and open source version of SSH. Today OpenSSH is the de facto standard for secure remote administration of Unix and Windows systems.

Open source provides a reliable implementation of directory services in [OpenLDAP](#), a collaborative effort to deliver a commercial-grade open source LDAP suite of applications and development tools.

For the classic middleware component, message-oriented-middleware, there are numerous open source options. Many are based on the Java Message Service (JMS) API, including [JBossMQ](#), [OpenJMS](#), [Open Source Message Queue](#) (OSMQ) and [JORAM](#).

Queuing systems connect applications asynchronously; for connecting applications synchronously and over different platforms, the technology of choice is, increasingly, Web Services. Microsoft’s .NET framework is one of the most prominent players. However, open source frameworks and libraries that use Web Services have popped up quickly.

The DotGNU project consists of three subprojects. [Portable.NET](#) aims to be a free and portable replacement for .NET. [PhpGroupWare](#), a multiuser Web-based collaboration suite, also provides a good collection of Web Services components, all of which can be accessed through XML-RPC so you can easily integrate them into Web Service applications of your own. And finally there is [DGEE](#), the DotGNU Execution Environment, which provides the core Web Service component of DotGNU and the functionality of accepting, validating and satisfying Web Service requests.

Another undertaking is the Apache Software Foundation’s Web Services Project @ Apache, which hosts several sub-projects, the most famous being Axis. [Apache Axis](#) is an implementation of the SOAP (Simple Object Access Protocol) submission to the World Wide Web Consortium; it is available not only for Java but also for C++. Based on Axis, many other projects have followed, fulfilling more sophisticated tasks like WSIF (invocation framework), addressing (implementation of the WS-Addressing specification) and WSS4J (the security needs of Web Services).

## Serverware

*Software that runs on servers – back-end, multithreaded, multiprogrammed computers accessed by multiple clients.*

The most well known category of open source software besides operating systems is server software. The [Apache](#) Web server powers 68 percent of Web servers worldwide, according to an August 2004 analysis from Netcraft, a U.K. Internet services company that tracks these figures. Thus Apache is used more widely than

```
int=None, REQUEST=None):
    rendered():
    red(self.pageType
    e().render(self
    for_cache=0):
    f.clearCache()
    e(*'prerender
    self.pageType()
    t(Permissions.Vt
    REQUEST=None):
    v cached render
    v_*
    v_cooked '*):
    v_cooked '*):
    v_blocks '*):
```

all other Web servers combined (both proprietary and open source).

The Apache Project, which includes many projects besides the Apache Web server, is the number two open source software project after Linux. Apache was started in 1995 by a group of people who wanted to contribute software patches to the HTTPD Web server written by the National Center for Supercomputing Applications; thus “a patchy” Web server was born.

The original Web server project expanded over time; in 1999 it spawned the Apache Software Foundation, which hosts several important open source projects including Jakarta, PHP, Struts and Apache XML.

Despite Apache’s success, many other open source Web servers exist, including [ACME http](#), [Roxen](#), [Jigsaw](#), [Amiga Apache Web Server](#), [EMWAC HTTP Server](#) and [TUX](#).

A close companion to Web servers are application servers. In a field of giants like BEA Systems, IBM, Oracle and Sun, open source application servers are making their mark. One of the first that appeared was [Tomcat](#), a servlet engine meant to be a standard

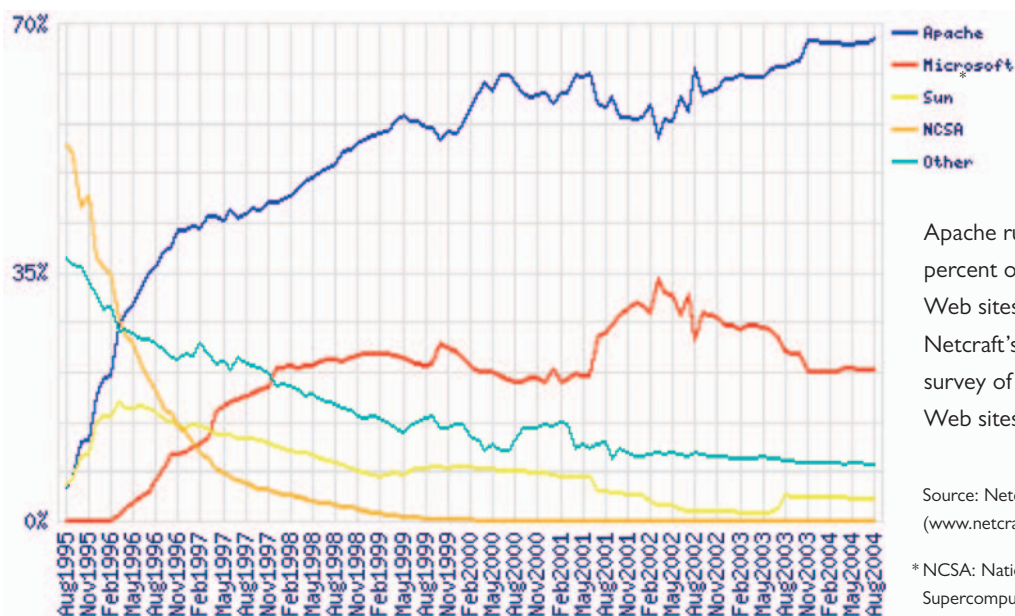
implementation of Sun’s servlet specification. Tomcat is regarded as the reference implementation today.

[JBoss Application Server](#) is the dominant open source Java 2 Enterprise Edition (J2EE) application server, with more than five million downloads. JBoss claims to be number three (behind IBM Websphere and BEA WebLogic), with a 25 percent market share. Although not formally approved by Sun, JBoss fully conforms to Sun’s J2EE-1.3 specifications and thus competes directly with the proprietary giants. Other open source application servers include [Jonas](#) (fairly stable and mature) and [Geronimo](#) (an up-and-coming competitor to JBoss but still in its infancy).

Note that it is beyond the scope of this report to cite all server areas where open source software is available; there are endless products for FTP servers, fax servers, mail servers and so on. However, it is important to examine one of the more important file sharing servers, [Samba](#). Samba is an open source server suite that bridges the gap between Unix and Windows environments. On Unix platforms, and especially on Linux, Samba provides seamless access to files and printers through Microsoft’s network protocols, like SMB/CIFS. When a back-office migration to Linux is considered, Samba

## APACHE RULES

### Market Share for Top Servers Across All Domains



Apache runs nearly 70 percent of the world’s Web sites, as shown by Netcraft’s Web server survey of over 50 million Web sites.

Source: Netcraft  
([www.netcraft.com](http://www.netcraft.com))

\* NCSA: National Center for Supercomputing Applications

products. As a versatile application database library, it provides persistent storage for DNS domain name servers, mail servers, network appliances, knowledge management, content management and document management.

Over the last five years, portals have become an important part of the enterprise infrastructure. In turn, there has been noticeably more portal activity in the open source community.

The most widely known open source portal framework is [Apache JetSpeed](#). Although stable as a framework and being used by some commercial vendors as the basis for their commercial portal server, JetSpeed aims to be a minimalist framework. At the other end of the spectrum is the [eXo](#) platform, an open source integrated application suite that provides several dependent, loosely coupled services including logging, persistency, XML processing and caching at the low end; a portlet container and workflow services at the middle level; and e-commerce and publishing services at the high end. Thus, eXo takes a full-function approach.

With the publication of the Java Portal Standard (JSR-168) in October 2003, portals have improved in terms of quality, features and standardization. The standard ushered in development at existing projects like JetSpeed and [Liferay](#), also an open source portal, to comply with the standard.

Another well-known open source portal is [PHP-Nuke](#). Thanks to the work of the open source community, there are more than 500 different modules that can be used to personalize a PHP-Nuke portal, including modules for weather, e-commerce, photo galleries, animated chat and video games.

Complementary to portals are content management systems. **OpenCms**, a prominent open source content management system, supports the creation and management of complex Web sites without requiring knowledge of HTML. A neat and simple editor with a user interface similar to well-known office applications helps the user create the content, while a template engine enforces a site-wide corporate layout. CSC has

15



```
port org.apache.  
public class Stand  
extends Contai  
implements Cor  
private trans
```

made two contributions to OpenCms: a translation of the online help from German to English and a correction that expanded OpenCms' Oracle database support.

```
public Stand  
super();  
pipeline;  
namingRes  
broadcast  
public void  
synchroni
```

Finally, a different kind of content management system that pervades the open source world is the [wiki](#). A wiki is a virtual shared space where documents can be authored collectively using a simple markup language and a browser. The speed of creating and updating pages is one of the defining aspects of wiki technology ("wiki wiki" in Hawaiian means super-fast). Generally, modifications are accepted immediately without review. An excellent collaboration tool, wikis are especially handy for projects involving people in dispersed locations (like this report's research team, which used a wiki). To date, the English-language Wikipedia, an open source encyclopedia on the Web, is the largest wiki, followed by the German-language Wikipedia. (See New Domains.)

### Manageware

*The myriad products used to administer and oversee the management of operating systems, network devices, serverware products and applications.*

Enterprise and system management is most often left to comprehensive systems like HP Openview, IBM Tivoli, BMC Patrol and CA-Unicenter. There is no sign of a comprehensive open source system management project that competes with the proprietary players.

Perhaps one reason is the lack of standards for consolidating information in data centers and for interaction between the different tools, servers and applications being managed. It was only in fall 2003 that some 40 system management vendors formed the [DCML](#) (Data Center Markup Language) Organization, committed to designing and developing an open standard to facilitate data center interoperability and better integration between tools. As the organization states on its Web site, DCML does for the data center what HTML did for content and IP did for networking: facilitate interoperability and make proprietary approaches obsolete by providing a vendor-neutral way to describe the data center environment and its management policies.<sup>10</sup>

Once standards are defined, proprietary and open source products will emerge, bringing more standard-

ized methods to the management of applications. Until then, vendors like Covalent Technologies will provide commercial products for Web application management that aim to "tame" the open source. The company supports major open source products in its [Covalent Enterprise Ready Server](#) including Linux, Apache, Tomcat, JBoss and MySQL.

One of the few open source products that provides server administration is [Webmin](#), a Web-based interface for system administration for Unix and Unix derivatives, including Linux. Although a valuable front-end administration tool for servers, Webmin is not – nor is it supposed to be – a replacement for an enterprise or network management system.

There are other individual open source system management tools that target a specific area. These tools are based on the current management standards SNMP (simple network management protocol) and JMX (Java Management Extensions).

SNMP, a relatively old standard, is supported by many open source products. Most of them are libraries enabling applications to provide information via SNMP. Only a few front-end tools exist for collecting, organizing and visualizing SNMP data. [Scotty](#), a Tcl/Tk-based management console, is one example providing a unified view of SNMP resources across a network infrastructure. And there is [OpenNMS](#), an open source network management system that provides SNMP-based functionality and can monitor services being provided on the network.

In Java server applications, JMX has become a standard for manageability. It is well supported by various open source products. On the application side, [MX4J](#) provides the framework for manageability. On the client side, [MC4J](#) provides a sophisticated management console with configurable dashboards and other features comparable to commercial product features.

The bottom line on manageware is that countless individual open source tools are available to manage, monitor and evaluate systems and applications in the IT infrastructure. However, there is no single open source project presenting a comprehensive, integrated approach. The individual tools are not well known,



## OPEN SOURCE SHORT LIST

Open source software permeates the software stack. This summary highlights the most prominent open source software in each area of the stack.

### CLIENTWARE

OpenOffice  
Mozilla Firefox  
Ximian Evolution  
GIMP

### MIDDLEWARE

Openadaptor  
OpenLDAP  
Apache Axis  
OpenJMS  
Hibernate

### PLATFORMS

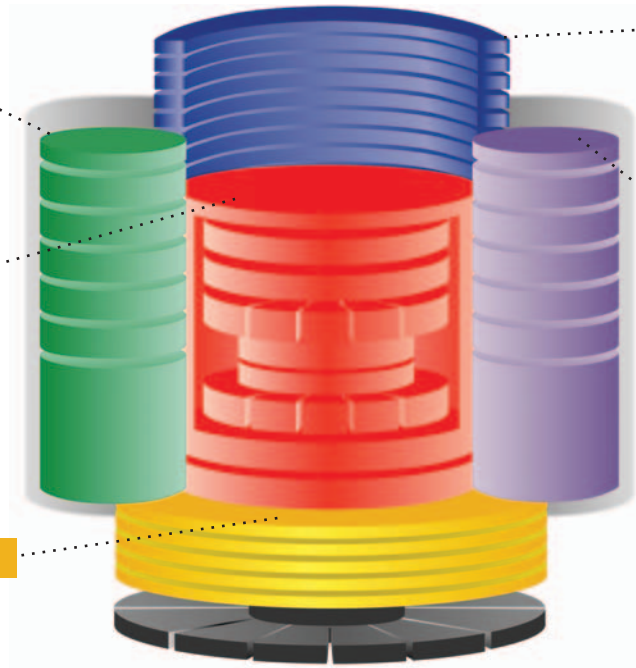
Linux  
FreeBSD  
Globus/OGSA

### MANAGEWARE

MX4j  
MC4j  
CVS

### SERVERWARE

MySQL  
Apache  
JBoss  
OpenCMS  
Jetspeed  
Samba  
Sendmail



Source: CSC with Flashmap Systems

and they would require considerable integration to be used in a management dashboard. We can expect to see more activity in open source for a comprehensive approach to system management and monitoring as this area emerges.

### Clientware

*Software that runs on clients or standalone computers.*

Linux is accepted as a server platform, but what about Linux on the desktop? Slowly, Linux is being perceived as a viable desktop operating system. What is important about clientware is that ordinary people – not developers or administrators – use it. Indeed, a key characteristic of a client – which includes PCs, workstations, handheld computers, PDAs and smart phones – is a user interface. Ease of use is paramount. This is important because Linux is not considered “user friendly;” it was originally written by developers for developers.

Thus, it was long doubted that open source could match what is generally available on the Microsoft Windows desktop, but times are changing. One of the earliest

non-Windows desktop operating systems, the **X Window System** (“X”), was the first open source project to have commercial impact. X was a fundamental building block upon which other commercial operating systems, such as Solaris and HP/UX, built their user interfaces. And without X, there wouldn’t be KDE, GNOME and other open source software.

X provides the foundation for a graphical user interface: drawing and moving windows on a screen, and interacting with a mouse and keyboard. X does not provide the user interface itself; that is handled by user software. Thus X-based environments can look very different.

KDE and GNOME, based on X11 (the current version of X), are two open source desktop graphical user interfaces for Linux. **KDE**, which is based on Trolltech’s Qt Toolkit, provides a desktop environment similar to Windows. Many innovations from KDE have been, or will be, adopted by Microsoft for Windows: desktop styles and themes, rounded windows, transparent windows.

# Are You Reinventing the Wheel or Using Open Source?

## *Soul of the Stack*

Making software development more efficient by using the right people and the right development tools is key to reducing IT costs. People associate open source software with infrastructure, but it is also important in the development toolkit. Often, reducing labor is seen as the only way to cut development costs, but working smart with the right tools can be as, or even more, powerful. This is where open source comes in, as the ultimate reuse library. Are your developers reinventing the wheel or using open source?

“Open source is the ultimate global reuse library,” observes Paul Gustafson, director of CSC’s Leading Edge Forum. CSC saved considerable development time in creating its Proactive Service Management solution by using open source software, which accounts for about one-quarter of the overall solution. Without open source software, CSC would have had to use much costlier, more time-consuming methods. (See Software Revolution.)

The history of open source is software by developers for developers.

Thus the development toolkit is rich in open source languages, tools, integrated environments and components. The key pieces in the open source development toolkit constitute the soul of the software stack, bringing to life the applications that run on it.

### Freeing the Languages

In many ways, the C programming language is the father of all open source projects. C boosted the GNU project and Linux significantly. GCC, the GNU C Compiler (later the GNU Compiler Collection), is the compiler that Richard Stallman wrote for the original GNU operating system. Because GCC was freely available and portable to many platforms, unlike other languages and compilers at the time, developers were able to create more tools for the GNU project. Linus Torvalds, the originator of the Linux kernel, based his entire development on GNU’s tools and GCC; many who contributed to Torvalds’ project used GCC as the basis for other development efforts. More languages followed, and many of the tools needed to work with them eventually became open source.

Inspired by C and other scripting languages, another free software language, Perl, surfaced in 1987. Perl was originally written by Larry Wall at NASA’s Jet Propulsion Laboratory. After its release, Perl caught on quickly and became the Swiss army knife of the system administrator. Perl became the first language at the dawn of the Web age for writing applications for the Internet.

One of the most prominent Web programming languages today is PHP. PHP started as a simple set of Perl scripts for monitoring Rasmus Lerdorf’s online resume (hence the name Personal Home Page/Forms Interpreter). When Lerdorf noticed that more functionality was needed, he wrote a larger implementation based on C that facilitated databases and helped PHP developers write simple and dynamic Web applications. Today, PHP 5 is a full-featured Web application development platform; several million sites have installed PHP, accounting for over 20 percent of the domains on the Internet. Countless open source projects – including portals, wikis, content management systems, groupware and forums – are written on PHP, with many more to follow.

Although not open source, Java is important to mention because it was the first language that could be used on any platform (write the application once, run it on many different systems). Other than C, Perl and PHP, Java probably has the most open source projects and support from the open source community of any programming language.

### Freeing the Tools

For a long time integrated development environments (IDEs) – where developers write, test and document the application – were rare in the open source world. Java developers had plenty of IDEs to choose from (most of them costly) and thus IDEs were often used inconsistently.

Things changed when Sun's **NetBeans** appeared on the IDE horizon and eventually became open source in 2000. (See Market Force.) A year later, IBM announced the release of a new IDE called **Eclipse**. Eclipse had undergone much development inside IBM before it was released as open source. Recently BEA Systems announced its intent to release **BeeHive**, on which its IDE is based, as an Apache open source project

to leverage BeeHive's power across the global development community and foster its continued development.

With several good open source IDEs to choose from, developers are consolidating on them. It is amazing how the pervasiveness – due to free availability – of these IDEs has boosted developer efficiency, brought consistency and ease of integration, and triggered new features almost every day. Developers use the tool, improve it, and share those improvements with others.

### Reuse and Reward

Armed with freely-available open source software, developers are witnessing a dream come true: applications can be assembled using parts from the toolkit rather than writing code from scratch.

In addition to languages and IDEs, there are numerous other parts in the toolkit, from complex frameworks to libraries of common and not-so-common functionality. Frameworks like **Struts**, **Java Server Faces**, **Cocoon**, **Spring** and **Webworks** give projects a kick-start and a common baseline. Unit testing libraries like **JUnit** boost

code quality overall. Sophisticated components like **Hibernate** and **Castor** make professional functionality freely available like never before.

It is amazing what can be achieved when the parts are assembled by a deft developer. Why reinvent the wheel when others have not only thought about but constructed the component? What makes the magic is that the code is freely available to be used, studied, improved and shared.

Boosting software development efficiency by using the right people and the right combination of tools from the soul of the stack will help managers win the cost game – and let the heart and soul of developers shine in unprecedented ways.

```
self.supportsC
return* CMFAwa
:
rendered = self
/* RESPONSE:
RESPONSE.se
return* rende
der*(self, cl
not* self.pre
lf.setPreRend
/* self.pageT
Render*(self,
lear_cache: s
ansaction(),n
etPreRendered
eclareProtect
arCache*(self
ly clear out
etPreRendered
asattr(self,*
latr(self,*
latr(self,*
REQUEST: REQUE
<DtmlIfNeeded
self.dtmlAllow
eclareProtect
<*(self, cook
see this page
ck.acquire()
lf._v_blocks=
lf._v_cooked=
/:
cklock.releas
uatePreRende
/* DTMLDocume
derMidsection
string.find(t
/: doc, discu
cept* ValueE
*return* t
```

**GNOME** (GNU Network Object Model Environment) is available for Linux, Unix and Unix-like operating systems and is the official desktop of the GNU Project. GNOME's usability, with an up-to-date look and feel, has made it a superior replacement for the established Unix desktop CDE.

A futuristic desktop environment is Sun's **Project Looking Glass**, a prototype next-generation 3D desktop environment. The project began as part of advanced development at Sun and was recently released as open source to gain insights from the community and to stimulate people's interest in the new technology. The project uses 3D windowing and other state-of-the-art techniques; it will be an interesting project to watch.

Sitting on top of the graphical user interface, the most widely used software package on the desktop is the office

OpenOffice has a mission to create, as a community, the leading international office suite that will run on all major platforms and provide access to all functionality and data through open-component-based APIs. Its major innovation is using compressed XML as its document file format (rather than a proprietary format).

For many companies, the office suite is still one of the most important applications in the information age. Compatibility with other office software is a major item being addressed by OpenOffice as well as other players like Sun with its **StarOffice**.

There is also open source project management software, namely **Open Workbench**, which is similar in functionality to the widely known Microsoft Project and provides robust project scheduling capabilities, including the ability to generate schedules based on resource constraints.



Project Looking Glass is an open source project at Sun Microsystems that explores a state-of-the-art 3D desktop environment.

Source: Sun Microsystems

suite, which includes word processing, spreadsheet, presentation and personal information management software. **OpenOffice**<sup>11</sup> is an open source alternative to the dominant Microsoft Office suite. OpenOffice is gaining momentum, with over 16 million downloads and countless installations from CD-ROM, according to its Web site. In contrast, Microsoft claims 300 million users and a 94 percent market share worldwide. (See Market Force.)

– is also critical. Coming soon from the Mozilla project is the popular open source e-mail application **Thunderbird**. *PC World* raved that “this surprisingly full-featured open-source program is a snap to use.”<sup>12</sup> Although still in its early stages at the time of this writing, Thunderbird provides a stable and promising e-mail platform with a sophisticated junk mail filter. Add the **Mozilla Calendar** to it and you have a nice personal information manager (PIM) environment.

But the office suite alone does not make a desktop complete; the browser is essential. Without repeating the story of the browser wars here, the browser space is teeming with open source versions such as **Mozilla Firefox**, **Nautilus** and **Konqueror**. Firefox is gaining increasing acceptance from former Internet Explorer and disappointed Netscape users. (The basis of Firefox is Netscape code, which was released as open source. See Market Force.)

Although the browser is key as the doorway to the Internet and network-centric applications, managing personal information – e-mail, calendar, address book, tasks, notes



```
print=None, REQUEST=None):
    rendered():
    red(self.pageType
    e(self.pageType).
    clear_cache=0):
    if .clearCache()
    e(*'prerender '*
    self.pageType).
    (Permissions.Vi
    REQUEST=None):
    y cached render
    ('*')
    _v_cooked '*')
    _v_cooked '*')
    _v_blocks '*')
```



# What's Different about Development with Open Source?

Software is software, whether open source or not. So how does the actual development process differ when using open source software versus proprietary software?

**Flexibility.** With open source software, developers are free to use as many applicable software components as possible. Developers are not constrained by license fees and budgets, which tend to restrict the use of proprietary components to a few comprehensive ones, resulting in overlapping and sometimes incompatible components.

Developers are also not constrained by proprietary protocols, which often require using one vendor's products, but instead can follow a best-of-breed strategy.

Nor are developers constrained by financial commitment. In a typical development environment, developers wouldn't dare discard ineffective components because of the investment made to purchase them. Open source software frees developers to try different open source components to see what works best.

**Support.** Open source support is pervasive and helpful, a big time-saver in development projects. Source code availability results in quicker problem

solving, a better feature enhancement process and collaborative learning. Code samples, snippets and tutorials help developers master new open source components.

Another support consideration is that the more expensive the proprietary software is, the fewer skilled people there are working with it (beyond the vendor). Thus the community the developer taps for secondary support (after the vendor) is limited compared to the much broader open source community.

**Innovation.** Open source software facilitates innovation because developers can experiment with components they would never use if they cost money (e.g., persistence frameworks, report tools, application servers). This is true with expensive as well as inexpensive software; in some companies, spending even \$50 has to be approved.

**Reuse.** Having access to source code increases the willingness to use open source software, encouraging reuse and weakening the not-invented-here syndrome that often pervades the development process.

**Quality.** The development process may be faster and smoother because developers can use available, tested

components rather than having to create and test components from scratch.

**Standards.** In general, open source software adheres more tightly to standards than proprietary software, providing more interoperability; sometimes open source projects create new standards. New standards are quickly supported by open source projects, enabling developers to try out leading edge software without creating it themselves. If the new technology is not suitable, the only sunk cost is time. Ultimately, one might still choose proprietary software, especially if it is perceived as less risky, as long as it complies with the standards and is interchangeable (e.g., a Java application server).

**Licenses.** This is a tricky topic with open source software, not to be underestimated. Organizations typically have a good understanding of proprietary software licenses but a less clear understanding when it comes to open source software. Open source software is often perceived to be free to use, period. In fact, open source software is licensed, and there are many different kinds of open source licenses. Organizations need to understand the provisions of the open source licenses they are using. (See Legal and Business Issues.)

business process functionality and off-the-shelf applications; however, there is movement in this direction.

Early signs are in Enterprise Application Integration, which includes messaging systems and adapter frameworks that help interconnect systems. One of these is [openadaptor](#), for improved application integration and e-business. (See Market Force). Astonishingly, there are even open source ETL (extract, transform and load) tools that integrate sensitive enterprise data from numerous sources and transform it into meaningful business intelligence.

Despite these open source capabilities, proprietary EAI platforms deliver more business functionality today, such as understanding SAP business objects or providing channel protocols like X.400 or formats like EDI. Thus, open source provides some of the building blocks but has yet to develop robust business-level functionalities. Of note: EAI systems delivering sophisticated functionality are typically far from low-priced, suggesting there is a ways to go before this class of software is commoditized.

Still, as will be shown in the Mission Critical trend, a number of industrial-strength systems are based on

open source and deliver core business functionality. Along these lines, some Enterprise Resource Planning systems are beginning to appear as open source software. One, [ERP5](#), features trading, invoicing, accounting, manufacturing, supply chain, stock, customer relationship management (CRM) and product design. The ERP5 project aims to not only release the source code but also provide all the necessary information (legal, social, theoretical) to allow anyone to introduce and implement the system, and the resulting process changes, in a small- to medium-size company.

When one looks at open source ERP or CRM systems like ERP5, [Compiere](#) or [GNU Enterprise](#), which are well on their way to becoming products, it is breathtaking to see the range of areas in which open source business projects cavort around the world. On the one hand, most of the projects are not comprehensive yet or haven't matured enough to be considered for corporate players. On the other hand, it is worth looking at these products for medium-sized organizations because open source definitely is an alternative not to be dismissed. This alternative will keep growing as open source moves up the stack to the higher levels, filling them with offerings from the tireless open source community.

## MISSION CRITICAL: Open Source Is Industrial Strength

The software stack is being put to work as open source software powers mission-critical projects. From financial institutions to governments to Google, it is clear that open source has arrived as an industrial strength tool. Although this is not the case for all open source software yet, there is no mistaking that open source is open for business.

That said, there is still some reluctance or even hostility to use open source software in a mission-critical context. Some argue that open source software is not mature or robust enough for mission-critical

environments – that it is not suitable for large-scale deployment and is only for small companies and zealots.

However, open source installations at major established organizations show that this not so. The tide is turning, as open source takes on mission-critical projects and mission-critical performance levels. The pieces are there – Linux, Apache, MySQL, Eclipse, Struts – and organizations are integrating the pieces into impressive capabilities. These are not pilot projects or ancillary activities. Open source is front and center, running the business.

## OPEN SOURCE AT THE HEART OF THE BUSINESS

At Deutsche Börse Group (German Stock Exchange) in Frankfurt am Main, open source software is widely used. CSC has been one of the partners with Deutsche Börse Group on its open source systems.

The consolidated architecture of all Internet applications, some of which are mission critical, is based completely on open source components in combination with a commercial content management system and a commercial database. Specifically, Documentum WCMS and an Oracle database were combined with a portal platform composed of Jetspeed, Apache and the open source search engine Lucene, which allows people to search the Web sites of [deutsche-boerse.com](http://deutsche-boerse.com), [www.clearstream.com](http://www.clearstream.com) and [www.entory.com](http://www.entory.com) (subsidiaries of Deutsche Börse Group). These open source components run on a JBoss application server with an integrated Tomcat servlet engine.

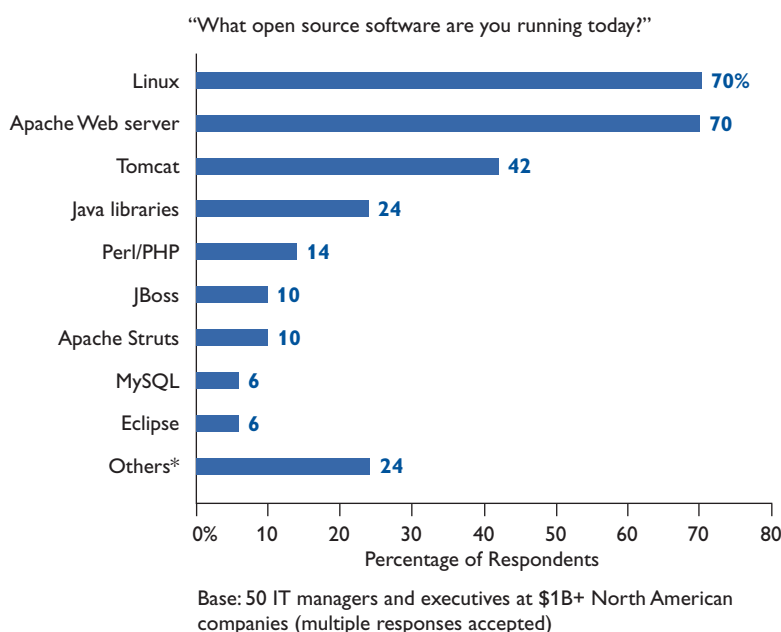
In addition, the reporting system that is used by Deutsche Börse Group to control the timeliness and quality of regular publications (e.g., annual financial

statements, quarterly reports) that companies have to deliver if they are registered at the German Stock Exchange in Frankfurt is based entirely on open source software.

“The selected open source components gave us the necessary functionality at the lowest initial cost and satisfied our requirements completely,” says Rolf Barth, head of e-Business at Deutsche Börse Group who was responsible for the organization’s Internet relaunch, which went live in 2003.

“All questions and requests related to the open source components could be handled by the project team. Sometimes the team had to spend some extra effort to find the correct documentation for a problem, or they had to fix a bug themselves without the assistance of the support team of a commercial software supplier. But these extra efforts would not have justified the extra costs for licenses and maintenance fees of commercial products,” Barth explains. “Since going live, the portal platform has been continually improved by our development team. After some minor problems in the beginning, we have reached an overall availability of 99.5 percent for our portals.”

## OPEN SOURCE IS MOVING INTO THE ENTERPRISE, LED BY LINUX AND APACHE



\*Includes 18 different open source products

Source: Forrester Research, Inc. “Your Open Source Strategy,” September, 2003.

The Danish Ministry of Finance is using a data exchange system made entirely with open source products to exchange account information between roughly 400 public institutions and the Ministry. The solution can transmit 1.5 megabits of data per second using a commodity Intel server running Linux.

Using open source components allowed the project team to create a solution that was architected to exactly meet the client’s needs as well as reduce overall complexity, according to Hans Jayatissa, head of eSolutions for CSC in Denmark and solution manager on the project. The open source software enabled a trim “mix and match” approach rather than the bloated “all or nothing” approach that often

```
erty.declareProtected(Permis
* *clearCache*(self,REQUEST
=====
Parcibly clear out any cache
=====
self.setPreRendered('*')
if hasattr(self, '_v_cooked
delattr(self, '_v_cooked
delattr(self, '_v_blocks
if REQUEST: REQUEST.RESPON
* *cookDtmlIfNeeded*(self):
if self.dtmlAllowed() *and
```

“The government was somewhat relieved to find that there were large vendors, like CSC, that would propose – and deliver – an open source software solution.”

comes from using a single vendor’s software suite, which tends to have more features than are needed.

“This was the client’s first use of open source software in a server environment,” Jayatissa observes. “This project was seen as aligned with government intentions to use open source software.

“The government was somewhat relieved to find that there were large vendors, like CSC, that would propose – and deliver – an open source software solution. Until then, open source software solutions seemed to be proposed only by smaller software vendors.”

By going open source, the Ministry got a simple, reliable and scalable solution that is easy to deploy, operate and maintain. In the long run, this translates into a lower total cost of ownership associated with the system. The solution, though created with open source components, is not open source itself. The solution is tailored code that is distinctly separate from the open source software used to create it; the Ministry owns the solution and has full rights to give or sell it to other government institutions or vendors (for maintenance).

The system is the forerunner of future open source projects at the Ministry. “The data exchange solution is an excellent start and can be used as a core component for data exchange in other areas of the Danish government,” notes Peter Henningsen, data exchange project manager for the Ministry.

At BlueScope Steel (formerly BHP Steel) in New South Wales, Australia, the company uses open source software to power a realtime process control and management information system for the Hot Strip Mill at Port Kembla. The system, which runs 24 hours a day, 7 days a week, 365 days a year, translates high-level steel product specifications into instructions for the process control hardware. Since the system went live in 2002, the mill

has broken long-standing production records. CSC provided round-the-clock production support for the first year of operation. The system uses Tomcat and Apache in addition to open source tools used for its development.

Open source is also hard at work in a mission-critical system for NASA’s Mars Exploration Rover (MER) Mission. A tool used to analyze data acquired by the Spirit and Opportunity rovers and plan their daily



At BlueScope Steel’s Hot Strip Mill at Port Kembla, Australia, the process control and management information system that runs the plant 24/7 is powered by open source software. The mill produces approximately 2.5 million tons of steel coils per year. In this picture hot steel bars are being coiled before being rolled through the Finishing Mill, where the final strip thickness is achieved. BlueScope Steel partnered with CSC to design and implement the new system with zero disruption to the plant. The system logs some three gigabytes of process data per day and communicates directly with over 10 controllers and measuring devices.

Source: BlueScope Steel



```

port org.apache.
blic class Stan
extends Contai
implements Cor
private trans
public Standar
super();
pipeline.d
namingReso
broadcaste
public void a
synchroniz

```



In this control room, called the Downcoiler Pulpit, the operator looks out over the downcoiler and post-coiler area of the steel plant. The monitors in front of him are his process screens, and above him are video monitors providing key viewpoints throughout the plant. Here the operator uses the open source-based system to monitor the quality of the final coiled product, record inspection results and log defect information.

Source: BlueScope Steel

activities was developed using open source software. This tool, the Science Activity Planner, is considered mission critical because a failure could jeopardize an entire day of operations for a rover.

Because of budget pressures, the development team turned to open source to satisfy as many system requirements as possible. The project has been a success; the open source components have contributed to the agility, high quality and stability of the system.

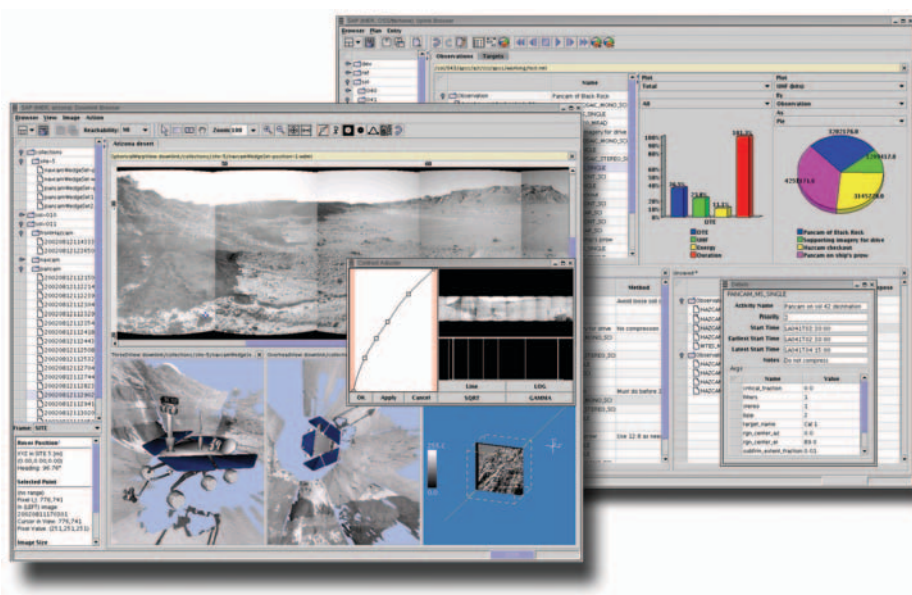
Back on Earth, the U.S. Navy is using an open source content management system by Zope Corporation to coordinate, document and track engineering changes for the broad array of equipment and parts the Navy manages. This is not a trivial task; all changes must be closely monitored and documented because any

change could have far-reaching consequences, be they for jet engines, ships or other sophisticated equipment (and those who use them).

## OPEN SOURCE FOR MISSION-CRITICAL PERFORMANCE

Open source plays a major role in applications requiring mission-critical performance. Open source is delivering on scalability, transaction throughput, reliability and manageability.

In terms of sheer size, the Associated Press' AP Hosted News is a massive service, providing AP news content for Web sites to some 600 affiliate newspapers and broadcasters worldwide. The AP uses the open source MySQL database to power AP Hosted News, handling hundreds of thousands of transactions a day.



NASA's Science Activity Planner, a mission-critical tool used to analyze data from the Mars rovers, was developed using open source software.

Source: NASA Jet Propulsion Laboratory



According to an article on MySQL AB's Web site:

...affiliate newspapers commonly divert more than 150,000 pages of content per day from their own Web site to AP Hosted News during an average news cycle, a number that can more than triple to as many as 500,000 pages per affiliate on a busy news day. Because affiliate news organizations rely so heavily on AP technology and content to retain reader loyalty, the underlying database in the AP Hosted News application must offer optimal performance and the ability to scale to support 11,000 concurrent users.<sup>13</sup>

Another application area with a large user base where open source is under consideration is the Australian Tax Office. In February 2004 the ATO, which serves millions of taxpayers, adopted an open source software policy for the first time, opening its Microsoft environment to alternatives such as Linux. There have been many government initiatives around open source software, as governments in China, Korea, Japan, Europe, Australia and the United States, as well as the United Nations, consider open source policy and options.

In Belgium, open source tools have been deployed for systems for the police as well as for the country's new electronic identity cards. CSC has been active in both efforts, using open source software for its technical merits. Commenting on the tools for the ID cards, which are being distributed to 10 million Belgian citizens, Marc Stern, security solutions group manager for CSC in Brussels, notes: "Using open source components was the only way to get the level of quality needed in the time frame needed. We actively participated in the development of some of the open source software, notably Apache, by adding some enhancements or missing functionality with the help of the community. These contributions helped ensure that the ID card system would run smoothly on most platforms, as the software has to run on almost any desktop computer used by a Belgian citizen."

Web search engine Google is perhaps the quintessential open source scalable project. The system is said to have grown to over 100,000 Linux computers spread over a dozen data centers around the world, serving up some 200 million searches on an average day.

Financial services firms, whose businesses depend on high transaction throughput and high reliability, are beginning to embrace open source. JPMorgan Chase has harnessed grid computing and Linux to lower costs and provide better internal IT service and flexibility. Its Compute Backbone, which consists of over 700 CPUs running Linux, manages a shared pool of computing resources for what had been seven discrete systems. These core business systems help traders assess and manage financial exposures such as interest rates, equities, foreign exchange and credit derivatives.

Several other Wall Street firms are using Linux in their data centers. One of these is Credit Suisse First Boston Corp., which reportedly processes some 60 million financial transactions daily. A few years ago Reuters ported its Reuters Market Data Systems to Linux. The system, which provides real-time market data and financial news, is used by major brokerage houses.

Web search engine Google is perhaps the quintessential open source scalable project.

Besides the financial world, another high-throughput environment is government elections. When 48 million people voted in the German parliamentary elections in 2002, the system for calculating the preliminary official results on election night used open source software (e.g., MySQL, Tomcat, JBoss). The system processed data from 80,000 polling stations, analyzing and displaying results in real time; the full analysis was completed at 3 a.m. the next day.

## FROM CUSTOM TO GENERAL BUSINESS APPLICATIONS

In addition to mission-critical custom applications for large organizations, open source is moving into mission-critical general business applications for small- to medium-size businesses. A good example of this is Compiere, an open source ERP solution for companies with revenues of \$2-200 million. Compiere's users include an auto parts manufacturer in the United States, a tire retailer in Germany, a metal parts maker in Brazil and a cable manufacturer in Singapore and China.

```
int=None, REQUEST=None):
    rendered():
    ed(self,pageType
    e().render(self
    ear_cache=0):
    f.clearCache()
    e(*'prerender
    self.pageType()
    t(Permissions.Vt
    REQUEST=None):
    v cached render
    '*)
    vcooked(*)
    cooked*)
    blocks'*)
    eprotected(Perm
    the*(self,REQUEST
    ear out any cod
    Rendered('*')
    r(self,*_v_cook
    (self,*_v_cook
    (self,*_v_block
    t: REQUEST=None
```

# Security: Secrecy Is Not the Answer

*“The open source community is in a better position to provide secure code than proprietary vendors because there are so many people reviewing the code.”*

Jason Arnold, program manager of CSC's H.E.A.T. security product

At first glance, open source and security appear to be an oxymoron, but in fact they are highly compatible: the very openness of the software ensures rigorous review and testing, bolstering security.

Still, one of the major concerns about using open source software is the perceived lack of security. The public availability of source code for the security components of a mission-critical system is an uncomfortable idea for those responsible for operating these systems.

This line of thinking sounds reasonable. If bad guys can access the source code, it is easy for them to find a weakness and compromise the system. Even worse, if they modify the code and succeed in getting those changes into an official distribution of the software,

they could easily insert back doors and Trojan horses into all systems using that software. This leads to the corollary: closed source software is more secure, as its source code is kept secret and only a few trusted programmers have access to it.

In addition to a malicious act, accountability is an issue. Who will fix a security problem if nobody is responsible for the code or for addressing the problem with the open source community? This is a nightmare for those running the system.

## Safety in Numbers

Fortunately, it turns out these concerns are not justified. In practice, we find there is safety in numbers: source code availability allows a large community of developers to inspect and review a system for security flaws. Key security components like security libraries, cryptographic algorithms, and authentication and authorization subsystems are attracting many developers to look at, review and improve the code for correctness and strength against security attacks.

In some cases monetary rewards are offered to further encourage developers to find bugs. After several high-profile flaws were found in

browsers, the Mozilla Foundation began offering \$500 for every serious bug found by security researchers in its open source software, which includes the Mozilla Firefox browser.

Developers are also driven by fame: finding a security weakness in a complex system is a challenging task that is recognized and lauded by peers. In widely used software like operating systems, or applications with a large public interest such as election software, developers are even more motivated to verify the system's security, for the impact of a security flaw is much larger — as is the tribute for finding and fixing it.

Finding a security flaw in a system is hard, even when the source code is available. But to create an exploit based on the flaw is even harder. Buffer overflow is a common security issue in server software written in C or C++ and has been the root of many exploits in the past. Buffer overflow is caused by the unwary use of static buffers in system calls from not checking the amount of data that is to be written into the buffer. If a malicious user succeeds in putting a specially prepared chunk of data outside the buffer's allocated memory space by having the program overwrite the

buffer, he can execute any code and thus compromise the system. Locating code vulnerable to buffer overflows is relatively easy, as it follows some common patterns. In contrast, using this for an exploit requires thoughtful preparation of specific input data, which is very hard to do in practice.

Having a large developer community distributed around the world leads to very fast turnaround times for security-related fixes compared to proprietary software. It is not uncommon to find a security fix to an open source product published on the Internet just a few hours after the problem's discovery. Moreover, large open source distributors like Red Hat and Novell/SUSE have set up mechanisms to collect and publish, in one place, security fixes related to all open source components these companies provide.

### **Closed But Not Necessarily Safe**

There is historical evidence that closed source software does not necessarily result in more security. In one prominent example, the voice encryption algorithms of the GSM standard – GSM is the world's most widely used mobile phone system – were developed secretly (closed source) but broken after being reverse-engineered.

Another example is the electronic voting system AccuVote-TS from Ohio-based Diebold Election Systems, the industry leader in U.S. electronic voting systems. In January 2003 the source code was inadvertently made available to the public from a file sharing (FTP) server. Shortly thereafter, a research team from Johns Hopkins University in Baltimore analyzed the code and discovered several significant security flaws, among them the ability to cast multiple ballots by a malicious voter. Diebold responded that the election process would prevent these flaws if everyone adhered to it. The example shows that the software itself is not proof against fraud, and since the source code is kept secret one has to trust the software vendor to fix it.

### **Open Source: Smart Engineering**

Computer security expert Bruce Schneier notes that in the cryptography world "...we consider open source necessary for good security; we have for decades. Public security is always more secure than proprietary security. For us, open source isn't just a business model; it's smart engineering practice."<sup>14</sup> That is why an open source algorithm was chosen for the AES (Advanced Encryption Standard) in 2002 to replace the aging DES

(Data Encryption Standard). The algorithm was selected after a three-year global competition led by the U.S. National Institute of Standards and Technology; AES is widely used in the commercial sector as well as in U.S. federal agencies. The competition required from the beginning that all submissions be publicly disclosed algorithms that were available royalty-free worldwide.

As Whitfield Diffie, co-inventor of public-key cryptography and chief security officer at Sun Microsystems, summed up in an article on ZDNet: "It's simply unrealistic to depend on secrecy for security in computer software. You may be able to keep the exact workings of the program out of general circulation, but can you prevent the code from being reverse-engineered by serious opponents? Probably not. The secret to strong security: less reliance on secrets."<sup>15</sup>

```
self.supportsC
return* CMFAw
:
rendered = self
/* RESPONSE:
RESPONSE.se
return* rende
der*(self, cl
not* self.pre
lf.setPreRend
/* self.pageT
Render*(self,
lear_cache: s
ansaction(),n
etPreRendered
eclareProtect
arCache*(self
ly clear out
etPreRendered
asattr(self,*
latr(self,*
latr(self,*
EQUEST: REQUE
<DtmlIfNeede
self.dtmlAllow
eclareProtect
<*(self, cook
see this page
ck.acquire()
lf._v_blocks=
lf._v_cooked=
/:
cklock.releas
uatePreRende
/* DTMLDocume
derMidsection
string.find(t
/: doc, discu
cept* ValueE
*return* t
```

In a testament to its rising popularity, Compiere has been downloaded over 630,000 times, making it the most popular open source business application to date. In February 2004, Compiere was voted SourceForge Project of the Month and was one of the top 10 most active projects on SourceForge, suggesting that this open source alternative for the mid market is catching on.

## GETTING ON THE DESKTOP

Open source has been slower to catch on in the desktop arena, where questions about functionality and compatibility (with Microsoft Office) loom large. For legions of office workers, office automation is their lifeblood – the mission-critical system they need to get their work done every day. If the system is not easy to use, or documents from inside and outside the organization cannot be shared and modified transparently, work cannot get done.

However, there are signs that open source is beginning to make inroads on the desktop. Just as Linux adoption rates have grown over time, we expect open office adoption rates to grow, albeit more slowly. Open office systems are creeping into large organizations through engineering departments. (See Sweet Spot.) Governments and businesses are asking serious questions about Linux desktops.

**“Just as organizations have gotten comfortable with open source software in their enterprise infrastructure, they are going to get more comfortable with open source on the desktop.”**

Open office technology is much more robust than even three years ago. The leading contenders are OpenOffice, an open source office suite (runs on Linux, Solaris and Windows); StarOffice, Sun's office suite (runs on Linux, Solaris and Windows); and Sun's Java Desktop System, a more comprehensive package aimed squarely at Microsoft Windows that includes StarOffice, Linux and a host of open source desktop productivity tools. The U.K.'s Office of Government Commerce has signed a five-year agreement with Sun to make the

Java Desktop System the desktop solution for the U.K.'s public sector organizations. Allied Irish Bank, one of the largest financial services organizations in Ireland, is planning to migrate 7,500 desktop users to the Java Desktop System. Developing nations including Brazil, China, India and South Korea are promoting the use of OpenOffice, which, among other things, is available in many more languages than proprietary office offerings.

In related moves, HP is one of the first major PC makers to offer Linux PCs with OpenOffice, which it began marketing in Asia in March 2004. The Open Source Development Labs, a non-profit organization advancing open source software, has launched a formal initiative to get Linux on the corporate desktop. The focus of its Desktop Linux Working Group, which includes HP, Sun, IBM, Intel, Novell, Red Hat and OpenDesktop.org, is to examine end-to-end Linux solutions as well as interoperability with other operating systems used in business.

“I think we are going to see an enormous movement to Linux on the desktop,” observes Bill Koff, vice president of CSC's Leading Edge Forum. “Just as organizations have gotten comfortable with open source software in their enterprise infrastructure, they are going to get more comfortable with open source on the desktop.”

Still, the office-desktop arena will be a tough nut to crack for open source given the entrenched base of Microsoft Office and Windows users. But the alternatives are there and gaining a toehold.

As organizations consider their IT strategy, they need to remember that open source offers a viable option, not just in the back office but in mission-critical environments. Consider the pros and cons. To be sure, different applications will have different requirements for scalability, throughput and reliability. For some, an open source alternative will fill the bill; for others, it will not. But clearly open source is doing the job today in many large-scale, mission-critical applications and will continue to do so as open source technologies are enhanced. Open source is open for business, and smart organizations will shop open source options for critical business needs.



## SWEET SPOT: Open Source Yields Targeted Savings

Although there are mixed reports about cost savings with open source software, organizations are realizing major savings in targeted areas, and these areas will continue to expand. As open source software moves into higher levels of the software stack, there are bigger savings opportunities. Increasingly, software and appliance vendors are embedding open source software in new products for cost savings and other advantages. (See Invisible Man.) And as more open source software is created and used, the potential for savings only expands.

The core of the open source savings proposition is no software license fees, reduced hardware costs from commodity hardware, and less unplanned downtime. Even if open source software is not deployed, it can be used as a powerful negotiating tool to lower proprietary software license fees and support charges.

Of course, costs must be examined comprehensively and on a case-by-case basis to get an accurate picture of the savings potential of open source software. Costs may decrease in one area and increase in others. What you save in license fees may be offset by increased technical support, training, consulting and other costs outside the bounds of IT. And unless it is a new project, the cost and disruption to transition from current application packages to open source alternatives may be prohibitive.

That said, significant opportunities exist for saving with open source software. In the quest to do more with less, organizations need to understand how these savings opportunities apply to their own operations.

Aaron Fuller, president of CSC's Defense Mission Engineering and Integration division, observes, "This is as true for defense enterprises as it is for commercial enterprises. Open source software and open systems architectures offer dramatic opportunities for improving the cost effectiveness of military systems.

"The post-Cold War post-9/11 international war on terrorism has focused military systems on joint and coalition operations in which weapons systems and troops from several nations are quickly assembled to form new military enterprises. The ability of these weapons systems and troops to work together is constrained by their incompatible computers and software designed as proprietary single-use solutions.

"Open source software and open architectures offer affordable implementations of interoperability that permit coalition military partners to work together as one unified military enterprise. Open source and open architectures are one of the most powerful components of this revolution in military affairs. The defense budgets of the United States and its friends and allies are not increasing significantly to pay for the high priority placed on interoperability. Open source and open architectures leverage technology to make interoperability affordable within tight budgets."

In both government and commercial environments, organizations face tight budgets and must maximize limited resources. Open source is an important option to consider. Three targeted areas for open source savings are the software stack, scale operations and software development.

### SAVINGS IN THE STACK

By examining the software stack from a cost point of view, organizations can find savings opportunities lurking inside many areas of the business. Of course, every business scenario is different; each organization must make its own calculations to understand where the savings opportunities lie. Here are some areas to explore.

#### Platforms – Linux in the Data Center

There is an opportunity to reduce platform acquisition costs by using Linux in the data center; not surprisingly, this opportunity varies by workload. Until recently,



Linux was only appropriate for applications that didn't need to scale to any large extent, or could scale horizontally (i.e., by adding servers). However, current versions of Linux now support sufficient vertical scaling (eight CPUs in a server is considered reasonable) to cater to most real-world workloads.

Horizontal scaling (assuming the application supports it) provides high availability by default, allows lowest-cost commodity servers to be used, and minimizes risk around sizing. Many application Web servers work this way today, adding more process threads to the mix to meet workload demands. (See Serverware on next page.) Linux is obviously a good option here as there are no horizontal scaling limitations. On the downside, horizontal scaling involves supporting multiple application and operating system instances (instead of one), has a connectivity overhead (multiple connections required to the network and external storage), and may have

software license penalties for any packaged software that is licensed on a per-server basis. This is a balancing act that needs to be assessed for each project.

That said, for many types of workloads horizontal scaling with Linux is a good approach. A horizontal approach can save big money in capital and operating system costs compared to vertical solutions on the market today. The ability to massively scale Linux horizontally without additional license fees can yield significant savings over the long run. This also puts downward pricing pressure on vertical solutions.

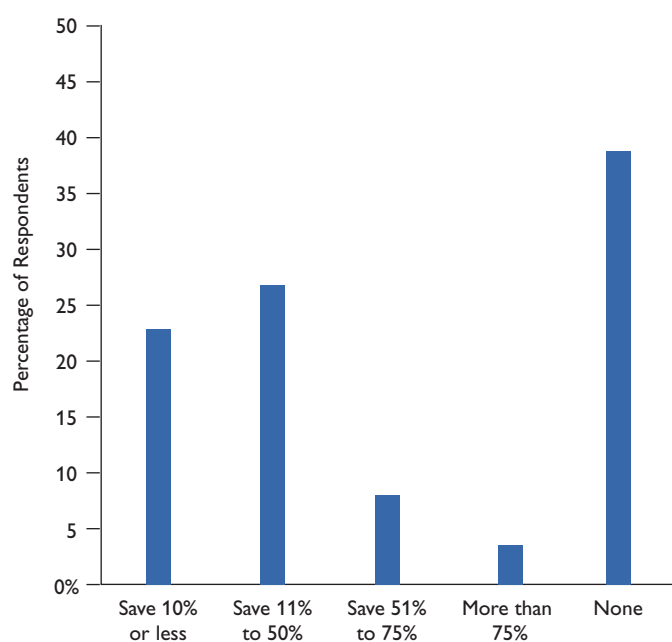
Vertical scaling is more technically straightforward as there is one application and one operating system instance to support, one connection to the network, one connection to storage, one set of licensing terms, and so on. However, there is significantly more risk around sizing. You may need to replace the server

with a larger one if it is undersized or if unanticipated growth occurs, forcing the organization to adopt larger servers with significantly higher unit cost.

Online travel services company Orbitz took the horizontal scaling approach (hundreds of Linux PCs) to undercut established competitors who were following a vertical scaling approach (mainframes). The horizontal, open source approach enabled Orbitz to create an innovative way to search for flights that was both faster and cheaper. Orbitz was able to bypass a key middleman in the ticket booking process; this, plus its lower IT costs, are at the heart of its business strategy to be the low-cost provider of the lowest air fares.

A large energy company offers another example of how a horizontal scaling approach can yield significant savings when coupled with open source software and commodity hardware. The company

## COST SAVINGS OF LINUX



In a survey of over 500 development managers conducted in December 2003, most (62%) expect some cost savings from implementing Linux, though the amount varies. More than 11% expect a dramatic cost savings of over 50%, while 23% expect a more modest savings of 10% or less; 38% expect no cost savings. The survey asked, "If your company were to move to Linux, how much cost savings would you guess there would be?"

Database Development Survey 2004: Winter, © 2004 Evans Data Corp.  
Source: Evans Data Corporation

figures it saved \$50 million in upgrading its high-performance computing center for advanced seismic research from specialized hardware running Unix to a cluster of commodity hardware running Linux. In the center, the company operates over 1,000 Linux (on Intel) systems, one of the most powerful commercial Linux operations in the world. The Linux systems are deployed using an open source grid engine from Sun. The company estimates that by moving to the Linux-based cluster it reduced its yearly software maintenance fees 93 percent, from \$3.5 million to \$250,000. The initial cluster deployment provided one teraflop of supercomputing power and has since been scaled to eight teraflops by simply adding more machines and processors.

With the cluster, processing that took 28 days now takes one day (with the ultimate goal of analyzing seismic information in near real-time). The time savings means the company gets a faster, clearer picture of underground areas offshore it is exploring as potential oil drilling sites. Given that it costs on the order of \$50 million to drill an offshore oil well, it is important to get the location right. Being able to produce the images faster – there is significant number crunching involved – means the exploration team can make adjustments and zero in on a candidate drilling site much faster. This capability could easily dwarf the \$50 million in savings attributed to open source.

If commodity hardware is already in place, savings with Linux are less clear, coming strictly from licensing and support costs (not hardware). Because support packages for Linux can cost more than licenses and support for Windows or Unix, organizations might opt for a non-commercial Linux distribution (\$0 price tag). However, they would forgo the integrity and security of the commercial distribution, which could be a costly tradeoff.

In terms of performance, for certain kinds of workloads Linux may yield better performance than Windows, a more feature-rich operating system that can slow performance. “In our experience, performance increases with Linux can be on the order of 20 to 30 percent, but again, this is very workload dependent,” emphasizes Tim Dooley, technology strategy manager

in CSC’s Global Infrastructure Services architecture team in the U.K.

## Serverware

**Web Servers.** In general, as you move up the stack costs increase, making the savings potential greater. In the area of serverware, Web server software is the low-hanging fruit for open source savings. Savings accrue from low to no acquisition costs, which can amount to millions of dollars in large organizations.

“If you are spending more than \$0 to acquire commodity Web server software, you are spending too much,” asserts Paul Gustafson, director of the Leading Edge Forum. Apache has become mature enough and popular enough – running two-thirds of all Web servers – to make this claim.

Of course, if the organization has sophisticated Web server needs, it may require a proprietary Web server that provides more advanced functionality (e.g., Zeus), is part of a wider integrated framework, or has independent software vendor (ISV) support. But for most organizations’ needs, Apache is suitable. Apache is an entrance strategy, best for new systems where there is no sunk cost.

Intel reports that according to analysts, more than 90 percent of Linux server revenue and shipments in 2003 were on Intel-based systems, and Linux server shipments were 25 percent greater than all Unix server shipments combined during the same time period.

Since Apache runs on several operating systems, there is no need to convert to Linux to realize Apache savings. However, Apache deployed on Linux yields the highest integrity and security, key factors for Web servers exposed to the public.

**Application Servers.** The opportunity for open source J2EE application servers is expanding from smaller, non-critical applications to mission-critical systems,

```
int=None, REQUEST
rendered():
ed(self,pageType
e().render(self
ear_cache=0):
f.clearCache()
e(*'prerender
self.pageType()
t(Permissions.Vt
REQUEST=None):
v cached render
'*)
v_cooked(*)
_cooked(*)
blocks!*)
```

```
eProtected(Perm
ie*(self,REQUEST
ear out any cook
Rendered(*'*)
r(self,*'v_cook
(self,*'v_cook
(self,*'v_block
IT: REQUEST.RES
IfNeeded*(self)
Ita(Allowed() *a
eProtected(Perm
itf, cook lock=th
his page's text
```

which used to be the domain of the established proprietary products like IBM WebSphere and BEA WebLogic. The stability and maturity of the open source application server JBoss, as well as its popularity among developers, make JBoss a valuable and low-cost alternative.

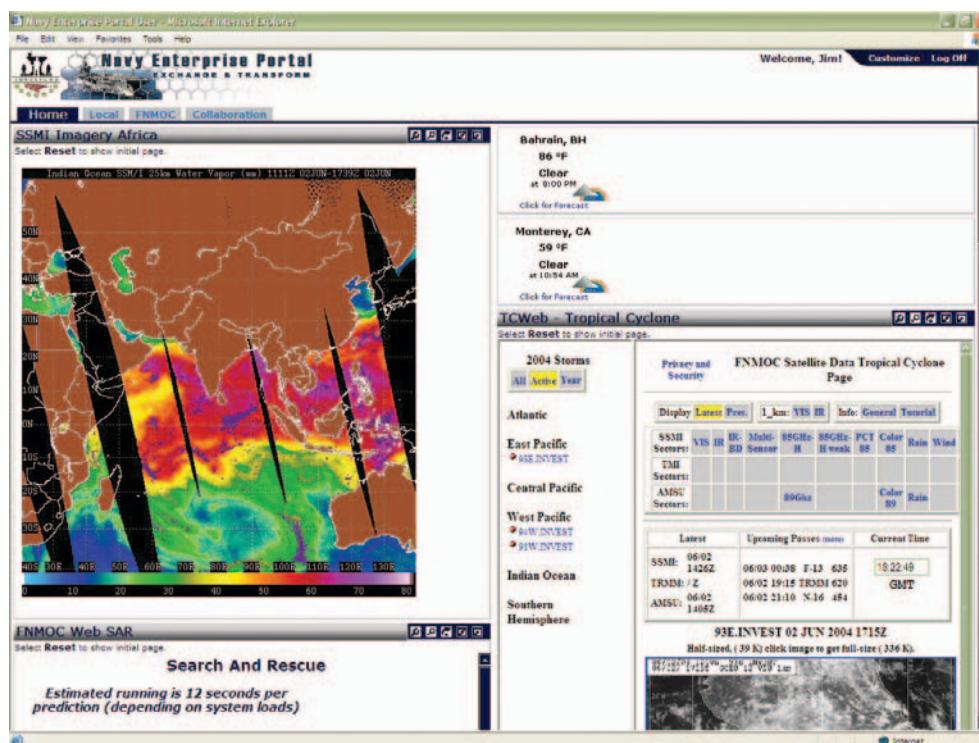
Many independent software vendors are using open source J2EE application servers as a low-cost development platform for their products (e.g., BroadVision and SeeBeyond). ISVs are supporting both open source and proprietary J2EE environments, though interestingly, their software often runs best in the open source environment. Expect this development trend to continue as a way to lower software costs.

Some large companies have already put JBoss on their list of recommended application servers. CSC has helped several clients develop and deploy mission-critical applications on JBoss. One of these was the data exchange system for the Danish Ministry of

Finance; CSC estimates it saved roughly 20 percent in development and deployment time overall by using JBoss and other open source software. (See Mission Critical.)

Another is a portal system CSC developed for the U.S. Navy that uses JBoss, Jetspeed (portal framework) and Struts (application framework). The Navy is intent on using open source software to reduce development costs and time; it estimates it could save over one million dollars in licensing fees, and that it saved six months of development time (initial development took 12 months). By not having to pay a per-seat license fee, which is common with proprietary portal software, the Navy was able to realize significant savings; these savings will only grow as the portal scales to more users.

Support was also a factor; with the open source community you can find out about new features and how to implement them, and get new features implemented by the community. This happened in the



This portal is an open source alternative to the current Navy Enterprise Portal, which is based on commercial software. The open source Navy Enterprise Portal, a full-service portal, saves development costs and time while providing enhanced functionality.

Source: U.S. Navy

portal project with a request to add portlet category filtering to Jetspeed.

Because of the open source software and standardized interfaces, the open source portal makes it quicker and easier to implement new functionality and applications than a similar portal the Navy is also using that was built with proprietary software and uses some proprietary features. The Navy's Fleet Numerical Meteorology and Oceanography Center (FNMOC), which developed the open source portal, has found the process of converting legacy applications, and developing new ones, to be much faster using the open source portal. Examples of converted applications include the Optimum Path Aircraft Routing System (OPARS), Search and Rescue (SAR) aids, and MyWxMap, an application that allows users to select areas of interest and download weather maps for those areas.

The open source portal was created by FNMOC to disseminate weather and ocean data to ships. FNMOC, the Navy's weather center, runs complex models and produces a variety of outputs including raw data about weather conditions, analyses to guide search and rescue operations, and optimal flight paths for Navy aircraft. In the past, requests came to FNMOC via e-mail or phone; now people can make requests from their browsers and have the information in minutes.

The FNMOC portal attracted the attention of Task Force Web, the Navy organization charged with implementing the Web-Enabled Navy initiative, as an alternative to the commercially-based portal it was implementing. The open source portal is now viewed as an attractive alternative to the commercial portal because of its lower cost and streamlined development path.

Bruce Gritton, FNMOC's CIO, said that the open source portal, and peripheral software such as the open source Navy Enterprise Single Sign-On (NESSO) and a subscription-based data distribution Web service called RuleBot, have been critical in fostering the rapid development required to allow FNMOC to modernize its internal processes and to become a ForceNET Development and Integration node.

As a testament to the portal's success, CSC is at work on a new Navy project that is the first project to be based on the open source portal. CSC, in partnership with the Navy, is implementing a Web-based application for monitoring compliance within the Navy with Information Assurance Vulnerability Alerts (IAVAs) issued by the Naval Computer Incident Response Team (NAVCIRT). Using the open interfaces provided in the open source portal, the team completed this project months earlier than would have been possible using the commercially-based Navy Enterprise Portal. Again, cost and speed were key factors in basing the project on the open source portal.

**Databases.** Database software carries a hefty price tag – several million dollars for an enterprise-wide license is not unusual – so this is an area of opportunity for significant savings in acquisition costs. Open source databases have existed for years, but organizations and application vendors have been reluctant to adopt them. One reason could be the business-critical nature perceived with the database: it houses the company's core business data. Others are the investment already made in existing databases, and the complexity of introducing yet another database into the organization.

That said, open source databases have improved and are gaining ground. (See Market Force.) Though open source databases are less likely to displace existing proprietary databases, they are a viable option in other areas, particularly new applications. In a March 2004 study on open source databases, Aberdeen Group reported over 10 million open source database installations worldwide, primarily in small- to medium-size business (enterprises or workgroups with less than \$25 million in revenue).<sup>16</sup>

According to Aberdeen, one of the key opportunity areas is decision support. Aberdeen identified what open source database users value most: the ability to handle wide variations in load and to process both large objects (or text) and relational data. Less important are ultra-high performance, extreme scalability in numbers of users or amount of storage, and administrative ease-of-use.

```
int=None, REQUEST=None):
    rendered():
    red(self,pageType
    e().render(self
    ear_cache=0):
    f.clearCache()
    e(*'prerender
    self.pageType(
    t(Permissions.Vt
    REQUEST=None):
    v cached render
    '*)
    vcooked'*)
    cooked'*)
    blocks'*)
```



```
port org.apache.  
blic class Stan  
extends Contai  
implements Co  
private trans  
public Stand  
super();  
pipeline;  
namingRes  
broadcast  
}  
public void  
synchroniz
```

New entrant vendors such as Metapa agree that customers are looking to unleash the power of open source and commodity hardware to tackle business problems such as decision support. Metapa recently brought to market an open source cluster solution that focuses specifically on solving business problems associated with data warehousing. The Metapa Cluster DataBase allows a cluster of commodity-priced hardware to be used to bring significant gains in performance and scalability to the warehousing problem. By building on Linux and PostgreSQL, Metapa believes that enterprises will now be able to deploy data warehousing and business intelligence systems at a fraction of the cost of traditional systems.

Typically these database cluster systems are deployed at the edge of the enterprise, which is where many Linux deployments are currently concentrated. Often the clusters are used where traditional systems are too slow and inappropriate given today's open source options and clustering with commodity-priced blade servers.

"What we are *not* seeing yet are deployments of these systems at the heart of the enterprise dealing with mission-critical applications and data," says Dave Powell, CEO of Metapa. "Enterprises are using these systems at the edge to learn about them and gain confidence with them. This is very similar to customer behavior during the migration towards client-server systems and, more recently, Internet-based systems."

Metapa has deployed its Cluster DataBase as part of a customer portal for managed firewall services provided by one of the largest U.S. telecommunications companies. The telco selected the open source-based system due to its scalability and performance, along with the obvious cost advantages that accrue from open source and commodity hardware.

## CLIENTWARE

**Desktops.** As mentioned earlier, open source software such as OpenOffice is being considered more and more for the desktop office productivity suite. OpenOffice offers savings on several fronts, but compatibility with proprietary systems – namely Microsoft Office – is a major obstacle to widespread adoption today, making any open source office suite a future opportunity for savings.

An open source office suite yields savings from lower acquisition costs and no forced upgrades. In large organizations, office suite software can cost on the order of \$300 per seat – or \$3 million in an organization of 10,000 people. That is real savings when going to acquisition costs of \$0.

A different kind of savings comes from not having to upgrade software every 18-24 months, which demands a significant investment of time and money. Upgrades are not only disruptive, but often they are not needed. By one estimate at least 40 percent of people in a typical organization can function effectively with office software that is "good enough" – i.e., that has basic functionality but not all the bells and whistles of a full-featured product.

However, a limitation that cannot be overlooked is the lack of a full productivity suite that is open source. Although OpenOffice is an open source alternative to Microsoft Office, there are not comparable alternatives for other tools such as Microsoft Visio. Moving to an open source desktop would require adopting alternative tools (maybe not even open source) for these capabilities, triggering significant associated costs, risk and disruption. Again, compatibility is an issue.

This issue will take time but it is at the top of the list. As OpenOffice addresses compatibility, the open source office suite will provide a major opportunity for savings. To be fair, compatibility is a challenge for both the open source community and the software vendor community. There are compatibility issues between open source software and proprietary software, as well as between various flavors of proprietary software (including different releases by the same vendor). Although compatibility issues cannot be denied, they will no doubt be overcome with time.

**Specialized Workstations.** With engineering workstations or specialized clients such as point-of-sale systems, savings and increased productivity can be achieved by switching from proprietary Unix workstations to Linux (on Intel) workstations, and by consolidating multiple machines onto one Linux box running multiple operating systems (using virtualizing technology).



A Linux engineering workstation project at a large telecommunications company is yielding significant benefits. In partnership with CSC, the company has been replacing 4,600 desktops and proprietary Unix workstations used by its R&D design engineers. In the first year of the effort, the company has eliminated over 1,500 secondary desktops and replaced a little over 1,000 workstations with Linux desktops – 35 percent of the total after the eliminations – and estimates they have saved \$890,000 in the process so far. The company estimates the three-year effort will yield \$4.1 million in total savings.

The company created a unique workstation consisting of a Linux host operating system and a Windows 2000 guest operating system (run using VMware), supplemented where necessary with server farms. This allowed engineers to design new products on Linux, existing Unix and Windows environments from a single Intel-based computer. This solution bridged legacy product designers with new products, providing work-force flexibility and agility.

The project is part of a broader effort to increase R&D efficiency and effectiveness. “Our goal is to boost designer productivity,” said one of the company’s IT leaders responsible for engineering R&D. “We wanted our design engineers to have a more flexible solution that wouldn’t require giving them a whole new desktop environment every time they changed projects.”

The project has been successful, boosting design productivity and converting die-hard Unix engineers into Linux fans. Said one engineer, “We designers love the reduction in load-build times. They’re 10 times faster, taking minutes on the Linux machine instead of hours.”

Observed the IT leader: “When people say open source is not ready for prime time, I disagree. In our R&D environment, it *is* ready for prime time.”

In an engineering environment, the major concern is application availability and delivery. Engineering applications tend to be highly specialized, and many do not support Linux. So the application may prevent realizing the full savings from open source, unless and until the application is ported to Linux.

This porting problem was solved at the energy company (mentioned earlier) by having the software vendor port its applications, which the company was using for sophisticated visualization, to Linux. The company is in the process of replacing 1,500 aging Unix engineering workstations with Linux commodity systems. The three-year replacement effort is expected to save \$25 million as workstations costing from \$30,000 to \$300,000 are replaced with \$20,000 boxes.

As at the telecommunications company, part of the savings comes from the consolidation of secondary office PCs, since some of the Linux systems also run Windows and its office software.

## SAVINGS IN SCALE OPERATIONS

Open source software presents a major savings opportunity in high-volume situations like manufacturing and services businesses, where savings accrue from millions of units involved.

When a manufacturer sells millions of PDAs or phones with embedded Linux, the savings multiplier can be enormous. Ditto for a services company that supports

“When people say open source is not ready for prime time, I disagree. In our R&D environment, it *is* ready for prime time.”

millions of desktops running open source software. The total cost of ownership (TCO) of the open source software to the manufacturer or services company is greatly reduced because the cost savings are felt across millions of instances; this translates to lower costs for the consumer or end user. CSC, for example, manages one million desktops and 40,000 mid-tier servers as part of its outsourcing business. In contrast, a typical large organization may have 50,000 desktops and 3,000 servers; TCO is constrained by this smaller base.

Single-function devices and consumer products such as bank ATMs, PDAs and mobile phones run well on embedded Linux, which requires minimal system resources and support and adds very little cost. Linux

```
port org.apache.  
blic class Stan  
extends Contai  
implements Co  
private trans  
public Stand  
super();  
pipeline;  
namingRes  
broadcast  
}  
public void  
synchroniz
```

helps keep the price of consumer products down and speeds overall product development because less new software development is required.

Motorola, for example, uses embedded Linux in products such as cellular phones to keep manufacturing costs down and to speed time-to-market by taking advantage of already available open source software it doesn't have to develop itself. (See Invisible Man.)

### SAVINGS IN SOFTWARE DEVELOPMENT

Significant savings can be realized from open source software development based on reuse and consistency. The telecommunications company mentioned earlier is moving to an open source collaborative model of software development in R&D to help drive efficiency. As it develops new products, the company, which designs hardware and software for enterprise and carrier markets, is aiming to consolidate more than 100 different proprietary platforms down to a small, standard set. The company expects to realize significant efficiencies in software development and field maintenance by eliminating duplicate functionality.

To do this, the company is bringing R&D together as a collaborative development community à la the open source model. This will enable more reuse and integration of software components across products.

The savings from collaborative development can be seen again and again. CSC jump-started the development process for its VP/MS Model Manager product by basing the product on Eclipse, the open source integrated development environment. (See Market Force.) The result was faster time-to-market by not having to reinvent the wheel, and reduced maintenance costs. After CSC contributed plug-ins from the VP/MS Model Manager project to the core Eclipse platform, CSC was relieved of having to maintain them; the Eclipse community takes care of that.

Another example of savings in software development is the Insight system CSC created for the U.S. Navy to manage the operating environment of a large-scale, distributed, real-time computer system. Given that the development schedule and other constraints would

not permit developing the solution from scratch, the team mined the open source collection in search of component solutions. The result was the integration of approximately 150,000 lines of open source code – ranging from tools and network services to configuration management and the graphical user interface – that saved millions of dollars in software development and met the 10-month development schedule.

### A CLOSER LOOK AT COSTS

In making the decision to use open source software, organizations need to understand several costs that are overlooked or not well understood: switching costs, legal costs, providing resources to the open source community, and the viability of the open source project long-term.

What does it *really* cost to switch your proprietary platform (or software suite) to an open source platform? It is fairly straightforward to determine savings from acquisition costs but much more complex to quantify migration costs, disruption to the business, training and ongoing support for the new platform. A detailed TCO analysis is needed. It must be noted that no matter how honest a person or organization is, TCO assumptions can be made to be optimistic or pessimistic; an objective TCO is hard to come by. The Total Cost of Ownership table on pages 40-43 offers guidance for doing as objective a TCO as possible.

Because open source uses a different support model based on a global community, organizations may decide they need to staff a person(s) to serve as the front line of support to the open source community.

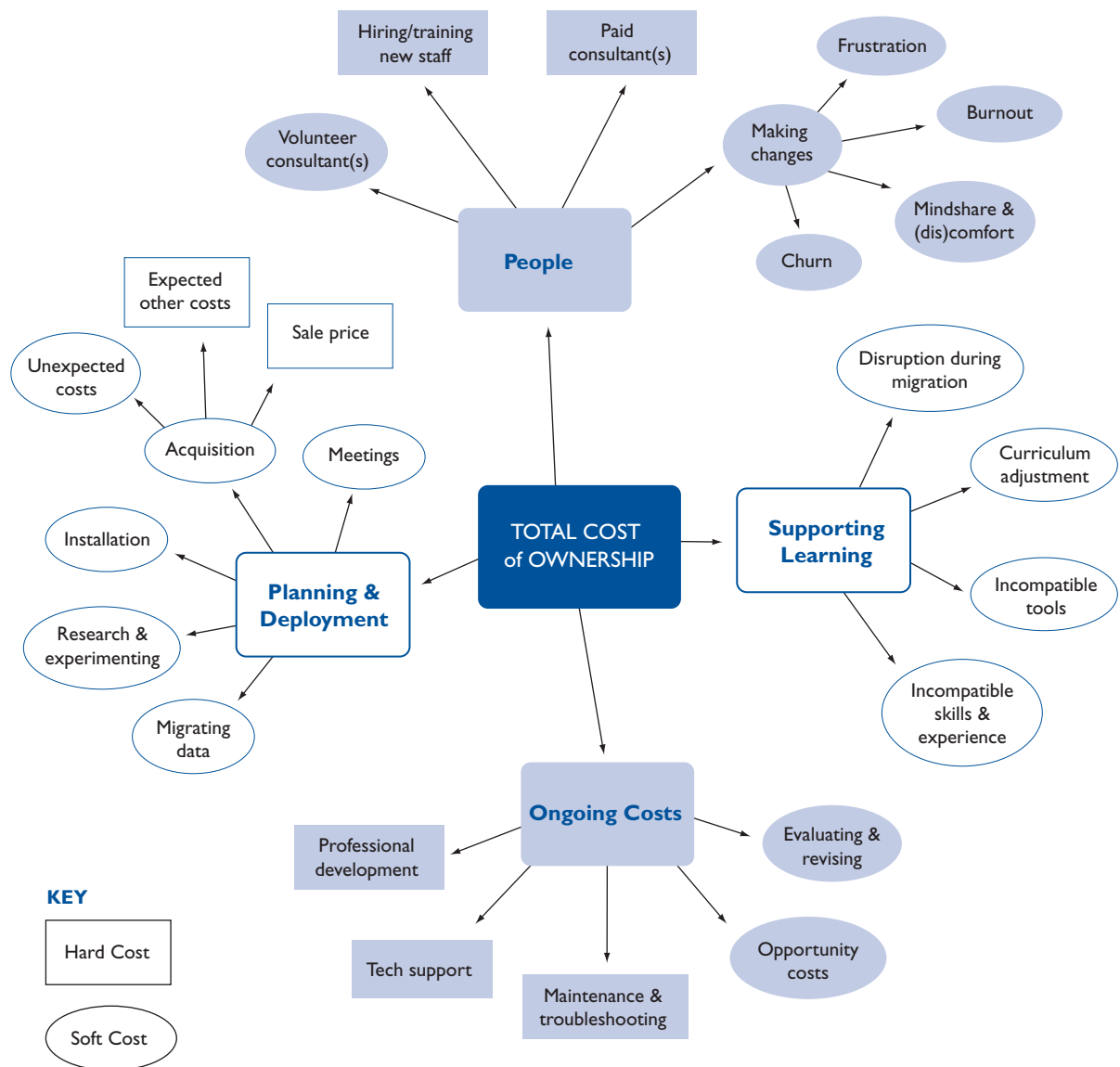
Finally, many organizations may ask: What is the long-term viability of the open source software I am using? There is a risk associated with relying on a volunteer community rather than a vendor for a software product (though some will argue the risk is no different, in fact, than relying on a vendor). Software fixes and enhancements may not be received in a timely or dependable manner; commercial business models for distributing and supporting open source software are still emerging. Also, what are the implications

of the SCO allegations or other possible intellectual property violations we don't yet know about?

Organizations need to take a hard look at where the opportunities for open source savings lie. The sweet

spots will vary by organization, but they are there. Managers need to get informed and get started on the open source agenda. Otherwise they risk leaving money on the table.

## TOTAL COST OF OWNERSHIP



Total cost of ownership for an information system is complex, involving not just acquisition and deployment costs but costs for support, training, skills development, disruption during migration and staff churn.

Source: Developed by the Northwest Regional Educational Laboratory, Portland, Oregon. [www.netc.org](http://www.netc.org) & [www.nwrel.org](http://www.nwrel.org)

# Total Cost of Ownership and Open Source Software

Total Cost of Ownership analysis varies from enterprise to enterprise. This table, based on several TCO models, summarizes the most important TCO components to consider when evaluating open source software. Enterprises need to take a comprehensive view of TCO and factor in other measures (e.g., technical advantages, security, vendor lock-in, long-term viability of the software, opportunity costs) to determine the full business value of open source software in their environment.

TCO Component	Open Source Software (OSS) Implications	Comment
<b>Hardware Costs</b> Purchase Price Hardware Maintenance	<p>OSS opens the door for more progressive use of cheaper commodity hardware (Intel running Linux) instead of proprietary hardware (RISC running Unix). For some workloads this can yield significant hardware savings, which are magnified when horizontal scaling is deployed.</p> <p>Savings also accrue from commercial applications moving to a commodity platform. Applications such as SAP, when moved from a proprietary platform to Intel/Linux, have yielded savings of 30 percent and more.</p>	<p>This discussion is very much a point in time and subject to change. Sun, IBM and HP have all significantly reduced the prices of their commodity RISC/Unix server product lines to more directly compete with Intel-based servers; this was likely fostered, in part, by competition from Linux and the fact that the vendors were seeing workloads drift from their platforms to Intel/Linux. The price per CPU for RISC servers and Intel servers has become very close in this commodity space. So from the simplistic perspective of CPU count, there are fewer opportunities for hardware cost savings.</p> <p>This represents a change from even two years ago. Then, and stemming back to the early days of Linux, there was a significant difference in costs between RISC and Intel platforms. But with competition and commoditization, that gap has narrowed considerably. Note: Even proprietary hardware vendors like Silicon Graphics are embracing commodity platforms, fueling the commoditization movement.</p> <p>In addition, some vendors have released a new lower-cost server line that competes even more directly against Intel/Linux, such as the V series from Sun. However, commodity Intel CPUs will continue to improve in price/performance at a far greater rate than RISC CPUs. Therefore, cost analyses should start comparing equivalent processing capacities, not number of CPUs, which would swing the balance much more in favor of Intel/Linux.</p>

#### TCO Component

#### Open Source Software (OSS) Implications

#### Comment

#### Direct Software Costs

Purchase Price

Support and Maintenance

License fees do not apply with OSS; however, costs are incurred for distribution (i.e., media fees) and support. Typically, proprietary software carries a high purchase price and comparatively lower ongoing maintenance costs. The reverse is true for OSS where a commercially supported distribution is adopted: there are low to no acquisition costs but significantly higher ongoing maintenance fees.

Since it is strongly recommended that enterprises purchase a commercial distribution of Linux (e.g., Red Hat or Novell/SUSE), Linux may be more expensive than Windows. It is, of course, possible to show savings with Linux if a non-commercial distribution is used, but costs will surface elsewhere (e.g., support).

Linux adoption can foster significant savings elsewhere, notably by moving from Unix-based software on RISC hardware to Linux on Intel hardware, which can reduce the number of CPUs required and thus license fees for software that is licensed on a per-CPU basis (e.g., Oracle). The increased Intel performance can require fewer CPUs for the same throughput. Software vendors could, however, minimize or eliminate this advantage – which today can be on the order of 30 percent savings – by changing their pricing structure.

Beyond Linux, savings can accrue from adopting other OSS (e.g., Apache, MySQL) instead of proprietary software. This yields significant savings in license fees. Enterprises can download the OSS or in some cases choose to buy a commercial distribution. Note that most OSS is not as volatile as Linux so enterprises may not need commercial distributions; the OSS is generally stable enough for enterprise use.

Overall, OSS has increased competition and put downward pressure on ISV and software vendor pricing.

continued on page 42



continued from page 41

TCO Component	Open Source Software (OSS) Implications	Comment
<b>Indirect Software Costs</b> Administration of Licenses	<p>Usually, it is not necessary to track the number of software licenses with OSS to ensure the legality of all instances deployed. However, you still need to know where the software is deployed for security and patching purposes. That said, OSS removes the burden of having to stringently monitor software deployment.</p> <p>It is important to have a strategy for which components should be used in the organization. This strategy must be evaluated regularly with proficient know-how of OSS components available.</p>	Software audits can be quite costly. It is no small task to create and administer auditing processes that allow you to know exactly, at any time, which user is working with what software across the entire enterprise.
<b>Staffing Costs</b> Project Management Systems Engineering/Development Systems Administration Vendor Management Administration (e.g., Purchasing) Training	Staffing costs mirror supply and demand. Early on, Linux professionals commanded a higher rate than Windows professionals because Linux professionals were in short supply. With the spread of Linux and Linux professionals, their fees are now comparable to that of Windows professionals.	Staffing costs depend heavily on the skills of the company's IT staff. For example, if a company has been running Windows as a strategic platform for a long time and has no Linux-experienced staff, it may be quite expensive to switch to Linux, though it may pay off eventually. A company's pool of available talent plays a major role in deciding what software suits the company best.

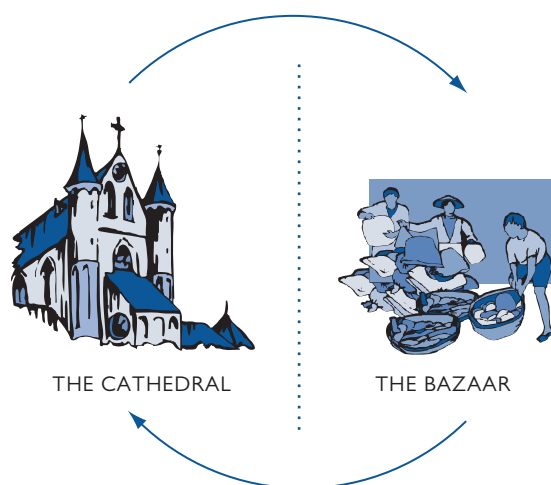
TCO Component	Open Source Software (OSS) Implications	Comment	
<b>Support Costs</b> Installation and Set-Up Troubleshooting Support Tools, Materials Patching Peer Support Casual Learning (manual, trial-and-error) Formal Training Application Development	<p>Generally, Linux is less of a target to malicious hackers than Windows and thus requires far fewer security patches than Windows, but probably slightly more than Unix. Also, because Linux is a leaner operating system than Windows is today, Linux requires fewer other patches but, again, probably slightly more than Unix.</p> <p>A commercial distribution of Linux is well-tested before its release and has tools for automatic patching. If you are not using a commercial Linux distribution, it is hard to keep up with patches because there are so many changes made on a rapid basis.</p> <p>Although installation and set-up historically have been more difficult with OSS, this is no longer the case with the major OSS products that come with easy-to-use installers. Troubleshooting costs may be lower with OSS (or not).</p> <p>Training costs may be higher depending on the OSS in question.</p>	<p>Compared to older versions of Windows, Linux environments have required less effort for troubleshooting and dealing with security patches, especially given the automatic patching that is available in a professional and efficient manner for commercial Linux distributions. Microsoft says this has changed with Windows XP and Windows Server 2003, though the company has released many patches for these operating systems.</p> <p>Tools exist for automatically patching Linux and Windows; CSC uses these tools, which are widely available to enterprises today.</p> <p>Community support in the open source environment is typically very good for the major OSS products. OSS newsgroups can often respond to problems faster than commercial vendors, though there is no contractual requirement for them to be that responsive.</p> <p>Some proprietary software products have become so widespread that they are the de facto standard (e.g., Microsoft Office). They are well-known by most people and thus generate few support costs, generally only when upgraded. However, equivalent OSS may cause disruption in displacing commonly used products.</p>	<pre> a.core;  .Container;  xt  rializable  log = LogFa  () {  new Standar tContainer( Notification  tionParamete  icationParam = parameter ; i &lt; applic e.equals(app cationParam  ;  ameter resu ationParamet py(applicat applicat ationParamet ameters = re  ("addApplica  ontainer ch  let = null;  ceof Wrapper galArgument ing("standa  (wrapper) ch et = "jsp".  { = (Wrapper) let != null d(oldJspSer </pre>
<b>Downtime</b>	<p>The modularity of Linux can allow a very lean build to be deployed, which in turn can enable more stability and thus higher availability than a Windows environment. (Note that Unix environments are also fairly stable.)</p>	<p>Unplanned downtime can be as much as 50 percent of TCO. Nevertheless, downtime is often not part of a TCO analysis because its costs are difficult to quantify.</p>	

## SOFTWARE REVOLUTION:

### Open Source Accelerates Development and Incubates New Ideas

The new software development norm is to collaborate and leverage: collaborate with developers in the open source community and leverage open source software. Rather than work within the confines of the organization and create everything from scratch, developers should use readily available open source tools and components to accelerate the development process, and tap the open source community for support and inspiration.

The open source community is an incubator for new ideas and innovation. The community socializes the idea, engaging the audience (developers) in dialog and development to create a concept (code) that addresses a particular question or challenge. The dynamic is just like a software development lab: pursue technically challenging or important ideas with an eye towards commercial application. But in the open source environment, the community can release an idea on its own, un beholden to corporate budgets and angel investors. This is revolutionary: community-based software development that accelerates and innovates the development process.



The lines are blurring between traditional software development (closed/"cathedral") and open source development (open/"bazaar"), fostering growth and innovation in software R&D.

There are four models of community-based development:

- leveraging open source in software development
- bootstrapping a new product
- extending an existing product
- forming a company to commercialize a product

#### LEVERAGING OPEN SOURCE IN SOFTWARE DEVELOPMENT

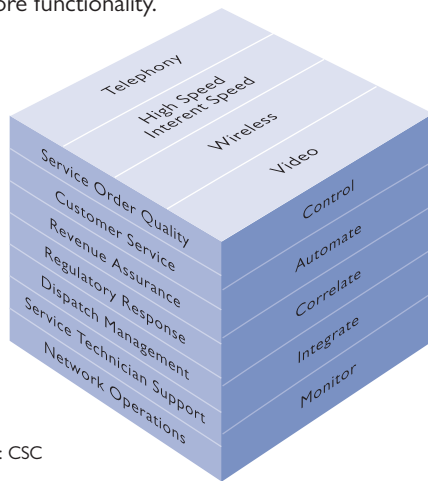
The leverage model is about using open source software to get the most out of your development effort. One example is CSC's Proactive Service Management solution, which used open source software for core functionality and for development tools. The open source software used was chosen because it worked (solved the problem) and was easy to use (people already knew how to use it or they could learn it quickly). In addition, it was free (no purchase costs).

PSM helps companies manage complex networks of remote devices in real time. Using PSM, the original client, a large U.S.-based broadband carrier, automated 90 percent of the telephony surveillance function, yielding millions of savings in cost avoidance, and reduced mean time to repair (MTTR) from 48 hours to four minutes, largely by automating 75 percent of a technician's research activities and making this information available to telephone service representatives, who can fix problems on the spot when a customer calls.

Approximately 25 percent of the PSM solution used open source software; there were three chief players: MySQL, PHP and Expect.

MySQL was used to store data for reporting purposes. Originally, MySQL was chosen as a temporary solution but it stuck because it performed well, and the overhead of going to a more heavy-duty proprietary database did not make sense. "This was a case where MySQL was 'good enough.' It fit our needs for a database exactly," says Bob Solis, PSM program director. In addition to storing data, MySQL was used for

CSC's Proactive Service Management solution, which provides a range of offerings for managing complex networks of remote devices in real time, uses open source software for core functionality.



Source: CSC

integrating various aspects of the PSM suite, scaling the application, and logging.

PHP was used for the front-end decision support system, delivering browser-based pages of reporting information that showed the health of every node in the system – vital data for operations company-wide.

In the developer's toolkit, Expect (an open source scripting language) was used to automate interaction with various telephone switches (e.g., AT&T 5ESS) and cable boxes. Many of these devices do not have standard application programming interfaces, or if they do they were cost-prohibitive, so Expect was used to create scripts to interact with the devices via their command line interfaces. This was an important breakthrough, because it meant the devices could be monitored and controlled automatically and proactively, eliminating what had been a manual effort and identifying problems before they occurred.

“Here open source software was used because there was no other way to get the job done other than to buy prohibitively expensive APIs,” explains Solis. The result was a creative, powerful innovation. Automation of the telephony surveillance function improved customer service and drove down business costs dramatically.

Other open source tools in the developers' toolkit included Apache, Perl, Log4J, CVS, Samba, Jitterbug

(defect tracking), DBTools (data management free-ware), Emacs (editor free-ware) and the GNU Compiler Collection (GCC).

Another example of open source leverage in software development is H.E.A.T., CSC's security tool for assessing information system vulnerabilities across an enterprise. H.E.A.T. (Hydra Expert Assessment Technology) was developed using several core open source components, which saved at least three months of development time in a 13-month initial development effort.

“H.E.A.T. would have been developed even if we hadn't used open source software, but it would have taken much longer and the product would not be as robust,” states Jason Arnold, H.E.A.T. program manager.

Since H.E.A.T. is a security tool, having inherently secure code is critical, which open source software helps ensure because so many people review it. (See Security sidebar on page 28.) Other reasons for shying away from using proprietary software were questionable functionality (you can't see the source code to know exactly how the software works and how well it integrates), licensing issues (CSC sells H.E.A.T. under its own license), and cost minimization.



H.E.A.T. (Hydra Expert Assessment Technology) is a security tool developed by CSC for assessing information system vulnerabilities across an enterprise. H.E.A.T., which uses many open source software components, performs detailed high-speed analyses of large, complex networks using a strategy that mirrors the unrelenting techniques of sophisticated computer criminals. From a development perspective, using open source software helps make H.E.A.T. not only more robust but also more secure.

Source: CSC

```
int=None, REQUEST=None):
    rendered():
    red(self.pageType
    e().render(self
    ear_cache=0):
    f.clearCache()
    e(*'prerender
    self.pageType()
    t(Permissions.Vt
    REQUEST=None):
    v cached render
    v_*):
    v_cooked '*):
    v_cooked '*):
    v_blocks('*):
```



Key open source software in H.E.A.T. includes: Apache, MySQL, OpenSSL, mod\_SSL (Apache interface to OpenSSL), “John the Ripper” (password cracking), Blowfish (encryption algorithms) and GD Graphics Library (report graphics). Using open source software enabled developers to concentrate on H.E.A.T. itself. For example, by using Apache, developers were able to concentrate on coding the core functionality of the product, which is vulnerability testing, instead of having to spend time writing a customized Web server.

H.E.A.T. runs on a network, probing specified devices for weaknesses and reporting its results. Organizations then mitigate or eliminate the vulnerabilities that H.E.A.T. finds. It is not uncommon for H.E.A.T. to probe thousands of devices in an enterprise, from servers to network switches. H.E.A.T. is much faster than comparable products; typical vulnerability assessments are completed in one week.

“SourceForge is my friend! As a developer, I go there first, even if it’s just to get an idea of how someone has done it.”

“Because of the reliability of the H.E.A.T. product, we are able to run assessments on devices in six continents and nearly every time zone without having to have anyone in the office,” reports Ken Ferguson, global IT security manager for the Huntsman Corporation, a leading chemical manufacturer based in Salt Lake City, Utah. “H.E.A.T. has allowed us to continually reduce our vulnerability over the two-year period we have used the product. We have demonstrated to business management our commitment to a secure environment with our progress using an objective tool that compares progress at over 100 sites spread across the globe.”

From a development standpoint, LEF director Paul Gustafson declares, “H.E.A.T. and PSM show that if developers aren’t using open source software tools, they are at a competitive disadvantage.”

Indeed, open source software should be the first line of inquiry. “SourceForge is my friend!” quips Arnold.

“As a developer, I go there first, even if it’s just to get an idea of how someone has done it.”

Open source software facilitates faster time-to-market and high-quality deliverables. “Having the tools available as open source software allows us to be productive right away and get the job done,” Solis observes. Because it can be readily modified and is strong and secure, open source software is key to custom development efforts.

## BOOTSTRAPPING A NEW PRODUCT

In the bootstrap model, existing open source software is the genesis of a new idea.

PasTmon, a passive transaction response time monitor, is an example of the bootstrap model. PasTmon is an open source utility that was developed by Graham Bevan of CSC. Motivated to solve a technical challenge – there were no such network traffic analyzers in the TCP/IP world – Bevan found a core part of his solution in Snort, an open source network intrusion detection system. Since Snort was distributed under the GPL, PasTmon, which incorporated portions of Snort, was also distributed under the GPL as open source software.

“I was playing around with Snort, doing research on security, and it dawned on me that I could leverage Snort to build the transaction monitor,” Bevan recalls. “I could also draw on the resources of the larger open source community for ideas.”

PasTmon was released on SourceForge in March 2001, just four months after Bevan began working on it. A similar proprietary product (there were none at the time) would have taken many more months to release. Today there are comparable products that can cost thousands of dollars per server per year. PasTmon is free.

## EXTENDING AN EXISTING PRODUCT

Open source can be used to extend an existing product. This is happening with products as playful as LEGO® MINDSTORMS™ and as serious as OpenCyc.

Toy maker LEGO, responding to hackers who were reverse-engineering its MINDSTORMS robot kits and creating open source development tools for them, supported the open source community by opening up

the product specifications, making the robots easier to program, and publishing advanced programmer kits and documentation. LEGO embraced the open source community to explore and extend its product, resulting in new functionality and a lively development community. (See Fun Factor.)

OpenCyc is another example of an existing product, Cyc, going open source to innovate further. Cyc is a common sense knowledge base that has been in the making since 1984, when development began at the MCC (Microelectronics and Computer Technology Corporation) under artificial intelligence guru Doug Lenat. Lenat and the Cyc project left MCC to form Cycorp in 1994. All along, Cyc was intended to go open source. However, it was felt that a certain amount of development had to be accomplished first; Cyc was being positioned as a platform for other applications to run on, and a substantial portion of that platform needed to be in place. In 2002 portions of the Cyc system were released as open source.

The turn to open source software development was two-fold: to grow the content in Cyc's knowledge base and to create new applications that run on Cyc. An underlying consideration was to help establish Cyc as the standard for knowledge representation, knowledge management and, in general, intelligent software applications.

Over the course of 20 years, some two million assertions have been created in Cyc's knowledge base. That number needs to be on the order of 200 million for Cyc to really excel, its developers say, and that's where open source software development comes in. Release the code to a global community to add many more assertions, and rapidly. It's like the Human Genome Project: it took several decades to complete just one percent of the work, but only one more decade to complete 99 percent of the work. That breathtaking achievement involved exponentially more people and better tools – exactly the course Cyc is following by going open source and employing more user-friendly tools.

One company that is focused on new applications for Cyc develops software for the packaging industry. This company is working with OpenCyc to develop an ontology for the packaging industry that would run

on Cyc. (An ontology defines industry terms in a clear and consistent manner, so that, for example, everyone has the same meaning for “customer.”) The idea is to release the ontology as open source to encourage industry-wide use, and to charge for the proprietary business applications the company is developing that draw on the ontology.

Overall, by going open source and accessing a broader development community, Cyc – and intelligent applications – has the potential to take off.

## FORMING A COMPANY TO COMMERCIALIZE A PRODUCT

This model is about open source jump-starting not just a new product or product extension but an entire company. This is the story of Intalio.

As a young and ambitious software engineer, Ismael Ghalimi had a vision for what would eventually become the commercial company Intalio. Ghalimi formed a hand-picked community of developers, already versed in open source software, to turn his vision into reality.

Ghalimi's vision was for a next-generation Internet-based enterprise application platform. He persuaded the developers, located across North America and Europe, that by working together they could make more of an impact against industry titans IBM and Microsoft than if they worked alone. Some donated open source software they had been working on, and modules were aggregated to create a number of parallel projects. The aim from the start was that the community was working towards the creation of a commercial product. This product eventually became the IntalioIn3 Business Process Management System, or BPMS. (CSC and Intalio co-founded the Business Process Management Initiative, BPMI.org, to develop standards for business process management.)

Although Intalio is a commercial enterprise, some of the open source software at the root of its products is available from ExoLab.org. ExoLab is the open source community that preceded Intalio. (Ghalimi founded ExoLab in 1999 and Intalio in 2000.)

ExoLab's contribution to Intalio was two-fold. Under license to Intalio, ExoLab supplied open source versions

```
int=None, REQUEST=None):
    rendered():
    red(self.pageType
    e().render(self
    ear_cache=0):
    f.clearCache()
    e(*'prerender
    self.pageType()
    t(Permissions.Vt
    REQUEST=None):
    v cached render
    '*)
    vcooked'*)
    ,cooked'*)
    blocks'*)
```

of existing commercial software, such as the Tyrex Object Request Broker. Using open source to replicate commercially successful software allowed Intalio to develop a robust enough infrastructure to have a viable business. Here, open source spurred development innovation.

ExoLab also supplied Castor, the XML-Java binding software. This was a genuine product innovation. Castor is at the heart of many commercial products.

Today Intalio's BPMS is considered cutting edge as a new category of software. By starting with an open source development process, Ghalimi and his community were able to validate innovative concepts, use components that would otherwise have to be sourced commercially, and develop new products quickly.

#### INNOVATING WITH OPEN SOURCE

Research and development is about new ideas; innovation. However, in the open source world innovation is sometimes challenged by what appear to be "me-too" products. Isn't that open source product just another version of an existing product? Our contention is that most open source software is innovative, though sometimes in less obvious or traditional ways.

Open source software is competing with nonconsumption, offering affordable functionality to those people or things otherwise left out.

Innovation, according to dictionary terms, is "the introduction of something new." In a software development environment, innovation tends to connote significant change – often a breakthrough.

But innovation is fickle; what is innovative in one environment is a commodity in another. E-mail may be innovative at the beauty shop but is a commodity in most corporations. An innovation can be a new product, like the first Mosaic browser, or features of an existing product, like tabs and extensions in the open source Mozilla Firefox browser.

Firefox's tabbed browsing enables the user to open multiple pages in one window as separate tabs. Its extensions facility provides additional functionality via small programs similar to plug-ins. These unique features are no doubt why Firefox is experiencing an uptick in market share (though it is still a dark horse in the browser race).

The plug-and-play extensions in Firefox enable the user to customize his browser quickly and easily, while relieving the primary development team of having to supply all new functionality. Instead, anyone can contribute an extension. Today there are over 150 extensions, including Mouse Gestures (customize mouse gestures to invoke commands), QuickNote (take notes with the browser) and SearchThis! (add Web sites to the right-click context menu, such as eBay, Wikipedia and Dictionary.com).

Thus, in addition to the extensions themselves, the extension facility is innovative because it fuels innovation.

Another open source product that may appear "me too" at first glance is the JBoss J2EE application server. Viewed cursorily, developing a J2EE application server is not innovative; the functionality is predefined by the J2EE specification, which every vendor must implement. JBoss started as a typical "me too" project, facing formidable competition from BEA (WebLogic) and IBM (WebSphere), who had already established their products.

Nevertheless, JBoss used a radically different architecture by implementing the application server as a loosely coupled set of services around a micro-kernel based on JMX (Java Management Extensions), which was a true innovation at the time, 2001. This meant developers could more easily change and adopt new technologies. That, coupled with the agility of the open source development model for rapid development, led to the success of the JBoss open source project.

JBoss again entered new territory by combining J2EE with aspect-oriented programming as the foundation of the next version of JBoss, released in 2003. Again, this step introduced a radically new view on enterprise computing and has attracted attention in the J2EE

world. Although it is not clear whether JBoss was the first to combine those technologies, JBoss succeeded in implementing and marketing them first. And now major players like BEA, typically the first-mover in the market, are following suit.

There are many other arenas that open source software is innovating, including: instant messaging (Jabber is opening up this proprietary space and innovating around standards), application development (Eclipse was awarded InfoWorld's 2003 Technology of the Year award for best application development tool), and, of course, the operating system. Linux is changing the economics of IT and therefore the corporation, powering the systems of large businesses and governments. (See Mission Critical.) It is also finding its way into numerous appliances, hand-helds and consumer devices, making products function-rich but affordable. (See Invisible Man.)

Back in 2002 a spokesperson for IBM, with its \$1 billion investment in Linux, told NewsFactor: "We believe that Linux is going to be the operating environment of the future because it fuels innovation like nothing else does."<sup>17</sup>

Many believe the open movement in general fuels innovation, not just in software but in every discipline

where community can be involved, from hard science to publishing. (See New Domains.) Ideas want to be free, and open source gets them out there and gets people working together and learning.

Open source also acts as a catalyst for competition, even if an open source product is not ultimately selected for use. The mere presence of open source alternatives is putting downward pressure on proprietary software prices, giving organizations a newfound negotiating tool. (See Market Force.)

But more and more, open source is the option being selected. And in many cases, open source is bringing computing to where it didn't previously exist, be that in developing nations or developing products. In this sense open source software is a disruptive innovation, as described by Clayton Christensen and Michael Raynor in *The Innovator's Solution*. Open source software is competing with nonconsumption, offering affordable functionality to those people or things otherwise left out.

Open source software development is agile; it enables people to bat around and release new ideas unencumbered. In its agility, open source revolutionizes software development and fuels innovation.

## AT YOUR SERVICE: Service Opens Up New Business Opportunities

There is a common perception that technical support is a serious shortcoming of open source software. However, support is proving to be fertile ground in the open source arena, rich in business opportunities for IT service and solution providers, software vendors, application service providers and others.

### COMMUNITY-BASED SUPPORT

Support is often a criticism of open source software. How does the software work? Who is accountable? Who takes ownership when problems arise? Support

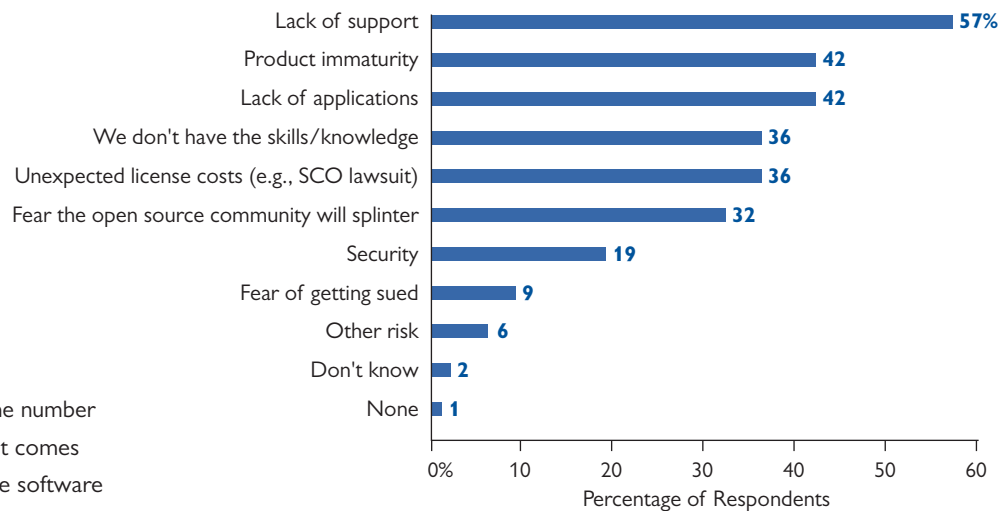
encompasses problem resolution, training, documentation, packaging, integration and other professional services.

In a traditional application development environment there are two support scenarios: the customer provides support (self-supporting) or the vendor provides support (typical customer-vendor relationship).

In some instances, there is a third scenario in which the customer and vendor share the "pain" of support.



## “WHAT ARE THE BIGGEST CONCERNS ABOUT LINUX AND OPEN SOURCE SOFTWARE?”



Base: 85 North American firms that use open source software (multiple responses accepted)

Lack of support is the number one concern when it comes to using open source software in the enterprise.

Source: Forrester Research, Inc.

This scenario is a bridge to the open source world because both parties have their eyes on the source code.

An example of the “pain-sharing” approach is CSC’s Virtual Customer Development. A leading vendor of financial services applications, CSC shares the source code of its insurance industry applications with customers so they can modify the code, but with the help of CSC. CSC incorporates customizations into the next version of the software, relieving the customer of this burden and strengthening its product line. (In the insurance industry, unlike most industries, applications are delivered in source code due to the high degree of customization required. These core applications are the “manufacturing line” of the insurance company, which defines its products by customizing these systems.)

In the open source world, there are new support scenarios involving a mix of customer, vendor, service provider and open source community. In the *customer-community* scenario, the customer provides self-support and taps the open source community as needed. In the *vendor-community* scenario, the vendor provides support and taps the open source community (open source distributors such as Novell/SUSE and Red Hat, or open source OEMs who embed open source software in their products, such as TiVo).

In the *service provider-community* scenario, the service provider uses open source software in a solution and provides support for the entire solution, tapping the open source community. That is how it works for CSC’s H.E.A.T. product for security vulnerability assessments, which contains open source software. (See Software Revolution.) CSC is the first line of support for its clients.

In a more collaborative scenario, the service provider collaborates with the client on who takes accountability for the open source software, and this party is the liaison to the open source community. Sometimes the open source vendor is also part of the mix if a commercial open source distribution is involved.

The community amplifies the support interactions already taking place by the customer, vendor or service provider. Although the community serves a number of functions (bootstrapping, innovation, camaraderie), its support role is critical in terms of product functionality and, ultimately, customer satisfaction.

### SERVICE PROVIDER AS TRUSTED MEDIATOR

And yet, many organizations are either not comfortable or not interested in interacting directly with the open source community. They want the trust, reliability and structure of an established commercial organization.

## Customer Value: What's Unique About Open Source

Red Hat identifies four unique characteristics of open source that provide value to its customers:

### **An active, large community of developers in which customers can participate**

The number of individuals developing in the open source community has been estimated to be anywhere between 250,000-750,000. Linux, as the largest open source project, features more than 30 million lines of code and has an estimated replacement R&D cost, using conventional development, of over \$1 billion. The value of this development community reaches customers through access to, and participation in, the ongoing development and roadmap of the software, in a way that isn't possible with proprietary software development models.

### **A large, global community of users**

Linux continues to gain market share in server and client markets. The size and global nature of the open source installed base results in the acceleration of innovation in open source software technologies and is driving widespread support for open source software among the largest enterprise software and hardware vendors.

### **The associated network effects of these communities, which benefit hiring, training, innovation and code quality**

As a result of large communities of development and use, IT managers are guaranteed a growing pool of open source skilled developers and IT support staff. The rate of innovation is greater due to larger numbers of developers and users associated with open source software, and the quality of the code is increasingly higher, again due to the large scale associated with open source communities.

### **Flexibility of code ownership and access**

Open source software enables timely access to source code and a constant stream of upgrades, security patches and source code enhancements. Open source software is the most rapidly evolving software ever, with patches and fixes constantly available.

Source: Red Hat

The free availability of open source software and source code opens the door for third parties to take on the responsibility of open source support. Gartner recognizes this trend, predicting: “By 2005, warranties and additional maintenance for at least the 100 most-popular open-source software products will be offered by commercial software vendors, service providers or insurance companies.”<sup>18</sup>

LEF vice president Bill Koff concurs: “Organizations using open source software will seek a trusted intermediary because they want accountability.” It is legitimate from a management perspective to have someone to blame if there is a problem you can’t or don’t want to solve. In traditional application development environments there are well-established project management workflows that mitigate risk. Issues like escalation, accountability, penalties and legal implications are well-defined; project managers and executives are experienced and comfortable with these processes.

**Taking on the responsibility for open source support, and mediating between the customer and the community, becomes the service provider’s new value proposition.**

However, similar risk-mitigation processes have yet to be created for open source software. As a result, customers may not be comfortable with open source because they perceive there is no one to go to for support. How can this accountability gap be addressed?

Enter the service provider, who can fill the accountability gap by providing front line open source support. This is a new role for the service provider, who in the past has deferred product support to the product vendor. Now, the service provider plays a more direct role in servicing the product, acting as a mediator between the customer and the open source community. For example, CSC can be the first line of support for open source software in large outsourcing deals. Taking on the responsibility for open source support, and mediating between the customer and the community, becomes the service provider’s new value proposition.

Clients, particularly outsourcing clients, will come to expect this. For example, open source was a key part of contract negotiations when the South Australian Government looked at renewing its outsourcing contract, estimated at \$1 billion, in 2003.

The service provider is well-positioned to take on open source support because providing IT service and support is its mainline business. Supporting open source software is an extension of the business, but with a significant twist: the service provider must learn to collaborate with the open source community rather than unilaterally pass problems to a proprietary vendor.

Although this is a critical issue for the service provider, in many cases it is transparent to the customer. The customer expects the service provider to deliver and support a solution, whether the solution uses open source software or not. The service provider is accountable for the entire solution and must have the expertise to support whatever software is implemented. In CSC’s experience, from the customer’s perspective the open source support issue is often nonexistent.

CSC has also found that support for open source software is often much better than generally assumed. According to CSC’s Hans Jayatissa, head of eSolutions in Denmark, where open source products are regularly used in solutions, “If a bug is found, it is more likely that we will report it to the open source project team rather than fixing it ourselves. Usually, the open source project team is able to deliver a new build with the fix included within 24 to 48 hours. This level of support exceeds any support you might expect from a commercial vendor.”

## ROLE FOR PURE PLAY AND SOFTWARE VENDORS TOO

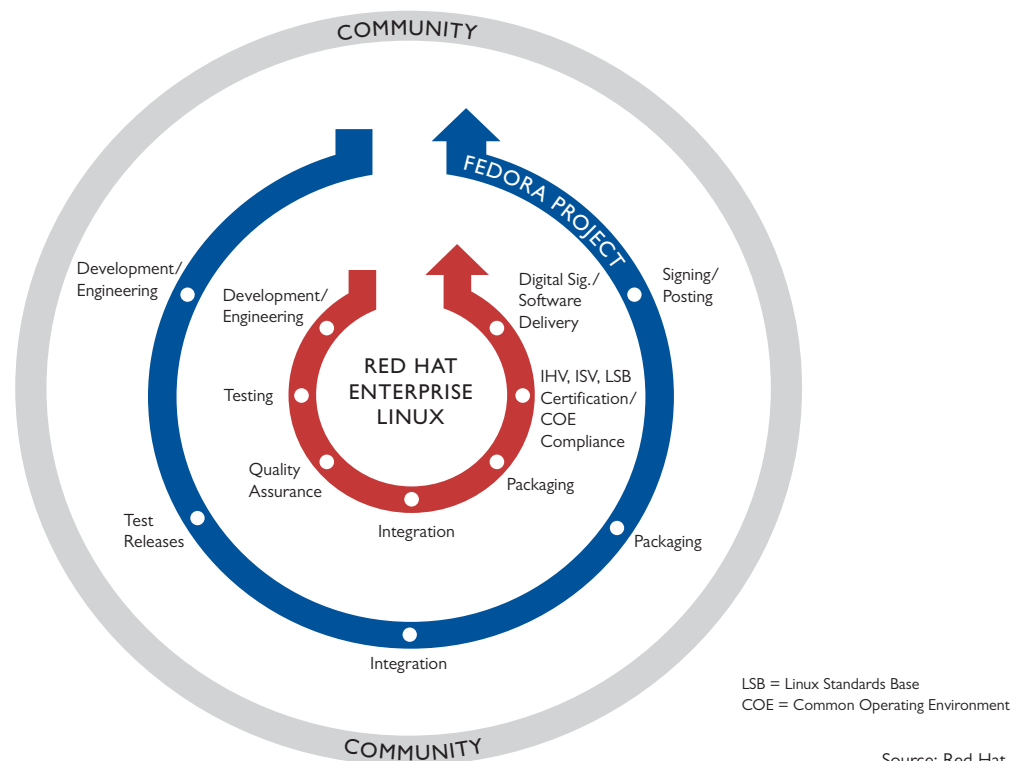
It makes sense that service providers embrace open source, not only from a customer standpoint – the demand is there – but also from a supply standpoint, seizing on the trend that open source is a viable service business. Open source companies Red Hat (Linux), Novell (SUSE LINUX), JBoss Inc. (application server), MySQL AB (database), Zope (content management) and VA Software (SourceForge) have built successful

businesses by providing enhanced software and a range of fee-based services including technical support, consulting, integration, training and documentation.

Red Hat, the most established of the pure-play open source companies, has made a successful business by taking aim at the unique needs of the enterprise. The company's Red Hat Enterprise Linux product line, for example, benefits from the innovations provided by an open source development model. Through Red Hat, enterprise hardware and software vendors have a standard platform on which to certify their products. Red Hat provides the necessary scalability and security

of open source software, making mission-critical Linux deployments possible. Red Hat helps to deploy, integrate, update, manage, train and support those enterprises implementing Red Hat Linux.

Red Hat assumes complete responsibility for bug fixes, maintenance updates, new features, system management technology and technical support. Red Hat provides a bridge to the innovations of the open source community. The company inherits the value of open source and Linux, and provides a convenient, accountable entity through which customers realize the benefits of Linux and open source.



Red Hat “productizes” Linux by starting with code from the open source community and adding service, testing, enhancements, and certification of independent software vendors (ISVs) and independent hardware vendors (IHVs). This is Red Hat Enterprise Linux. A new version is released every 18 months, with ongoing bug fixes and security errata provided through the Red Hat Network. An update that rolls up the bug fixes and security errata is released every three to six months. Each version maintains binary compatibility through all the updates. Each version is supported for five years (new hardware platforms are certified on each version

for the first three years). ISVs and IHVs are certified on each new release.

The Fedora Project is a Red-Hat-sponsored and community-supported open source project. It is a proving ground for new technology that may eventually make its way into Red Hat products. The Fedora Project is open source code that Red Hat releases at no charge; there is a new release every four to six months. The Fedora project is used by end users, developers and researchers who want to be on the leading edge of open source development.



“This is not only about bringing open source software to market to solve real business problems, but also recognizing the power of the network effect on the open source movement,” says Michael Tiemann, Red Hat’s chief technology officer. “Metcalf’s Law, which says that the value of the network increases by the square of the number of users or devices connected to it, applies to open source. The more developers who participate, the greater the power of the open source community to continue to generate new ideas, ultimately creating a win-win proposition for all.”

The result has been a winning combination. In reports for the quarter ending February 2004, Red Hat reported having sold 87,000 subscriptions for Red Hat Linux while Red Hat’s nearest competitor sold 3,800 subscriptions in a similar time frame. This has given Red Hat significant leverage with ISVs and hardware OEMs. Offering enterprise hardware and software vendors a standard platform on which to certify their products, Red Hat’s software is used by over 900 certified ISV applications, further strengthening the software’s credibility and use in the enterprise.

Novell is another case of a vendor building a business on open source. Once considered moribund, the networking company is revitalizing itself with open source, having acquired leading enterprise Linux distributor SUSE LINUX in 2004. Novell, a classic proprietary enterprise, is turning to open source to breathe life into the company – a testament that open source is a serious service business. (This could be the “Opensoft” move of the decade.)

“The more developers who participate, the greater the power of the open source community to continue to generate new ideas, ultimately creating a win-win proposition for all.”

In a more recent move, HP began providing technical support for JBoss and MySQL in June 2004, showing that established proprietary vendors are well positioned to play the open source service provider role. As well, HP, Sun, Novell and other vendors are providing

indemnity for their Linux customers against potential SCO lawsuits, illustrating that there are many sides to service and reinforcing the vendor’s role as a trusted provider. (See Legal and Business Issues.)

## NEW OPPORTUNITIES FOR ASPS

Application service providers are also getting into the open source act. ASPs are seizing on a viable business opportunity: applications with zero software costs around which to leverage a service business.

The open source proposition for an ASP is a business free of license fees. By hosting open source applications, the ASP does not incur any license charges for the applications being hosted. This lowers the barrier to entry for an ASP, opening the market to small companies and start-ups who may be rich in expertise but not cash.

For example, start-up Workspot provided a Linux desktop platform and applications as an ASP. Users could access the platform, hosted by Workspot, via their browsers for \$9.95 per month. Workspot’s vision was to provide a personal, infinite machine – an eternal desktop available remotely from anywhere. That could only be done using open source software. Though Workspot disbanded after five years (in 2003), it was a very forward-thinking ASP strategy.

In addition to start-ups, traditional ISPs can leverage open source software to expand into the ASP realm. ISPs, with their robust technology infrastructure, are well-positioned to do this. What’s needed are more thin client environments – kiosks, hand-helds, browser access to full applications – to stoke the flame of the ASP market.

Clearly, open source business opportunities are moving up the stack as open source commoditizes software infrastructure and applications. Service opportunities expand from professional services related to application delivery, to providing applications and services directly to end users. Open source guru Tim O’Reilly calls the latter an example of a “hidden service business model” for open source.<sup>19</sup>

O’Reilly maintains there are several hidden service business models, ISPs being another. For example,

O'Reilly has stated that UUNet, not Red Hat, is the greatest open source business success to date. UUNet (which became the Internet unit of MCI) built a successful billion-dollar ISP business on open source software, dwarfing Red Hat, a roughly \$100 million business.

### SERVICE IS KING

Having customers use open source software for free and pay for technical support is the first level of business around the open source concept. The next level is having

customers pay for the open source software or applications as a service delivered by an ASP, along with support.

Either way, service is king, and rightfully so. Organizations can expect to pay for professional services and support to make the best use of open source software, the same way they need the professional services of attorneys to make the best use of publicly available laws. Enterprising service providers will be “at your service” for open source.

## INVISIBLE MAN: Open Source Is All Around Us

What do watching your favorite movie on your TiVo, surfing the Web wirelessly from your terrace, and listening to MP3 music in your car have in common? All these technologies use open source software “under the hood.”

Embedded open source software is a rich market, driving the adoption of open source software further. An increasing number of network appliances, consumer electronics products and gadgets, and even mobile devices for the military are powered by Linux and other open source software hidden inside.

As open source software embeds itself everywhere, we are witnessing an engine of growth in information technology not unlike the railroads, a comparison made by economist Brian Arthur. According to Arthur, when technology embeds itself deep in the economy, it becomes an engine of growth for the economy. This long technology “buildout” happens after an initial boom and bust with the technology (recall the dot-com bubble and burst) and can last decades.<sup>20</sup> During the buildout, the technology becomes part of the fabric of life – even invisible – and just works.

### FROM ITS ORIGINS...

When Linus Torvalds began developing Linux, he never expected Linux to run on anything but a 386-based PC. However, the dynamics and power of the open source

community proved him wrong. Shortly after various projects began porting the original Linux kernel to large servers and workstations, several developers aimed in the opposite direction: bring Linux to small-scale hardware. Following the early success of projects like ELKS (Embedded Linux Kernel Subset) and uCLinux (micro-controller Linux), many free and commercial distributions of embedded Linux systems appeared on the market.

Today embedded Linux provides the software foundation for many consumer products, offloading the burden of software development from the manufacturer, delivering reliable functionality, providing a new level of robustness and capability to innovate on, and keeping costs down – especially critical for consumer products and network appliances.

### ...TO THE NETWORK RACK

Firewalls, routers, modems, wireless local area network access points, and other network appliances are ubiquitous in today's computing infrastructure. Most of these unimposing black boxes require a sophisticated software system under the hood to be able to do their day-to-day operations. Thus it is understandable why Linux is the number one choice of embedded operating systems by network hardware manufacturers, most of whom are located in Asia. In a survey conducted in October 2003 by *EE Times – Asia* and Gartner,

depending on the region, 40-50 percent of the vendors reported using embedded Linux as the base for their firmware.<sup>21</sup>

The massive impact Linux is having on the embedded software market is reinforced by the actions of Wind River Systems, a leading provider of tools and platforms for embedded software development, to fully support Linux and to create an open IDE built on Eclipse. Early this year, Wind River announced it was teaming with Red Hat to develop an integrated embedded solution based on Linux and Wind River's development tools.

The powerful basic framework that makes up Linux is not the only reason network hardware manufacturers rely on the open source operating system. Because Linux has sophisticated networking capabilities itself, such as a full-fledged packet filter and firewall subsystem, it is an ideal base for building network appliances.

## Open source software could be the breakthrough needed to finally converge computing and TV.

An innovative combination of a network appliance and a business phone is the ZIP 4x4 IP Telephone by Zultys Technologies. At first glance, the ZIP 4x4 looks like an ordinary analog phone, but in addition it provides voice over IP functionality (Internet phone) and has a network switch, firewall and Internet router built in. Like other network appliances, Zultys' product is based on embedded Linux. Bridging the gap between analog phone and digital Internet, this product is a major step towards the consumerization of open source underneath; the open platform helps make such a phone part of the standard business infrastructure of the future – pushing Linux and open source along with it.

### ...INTO THE LIVING ROOM

One of the earliest consumer products that used Linux as an embedded operating system was TiVo, a set-top box that digitally records television programs. Released in 1999, TiVo was the first commercially available digital video recorder (DVR).

The ZIP 4x4 integrates a business phone with an IP phone and is based entirely on open technologies, including Linux. The phone is compatible with any IP telephony system using SIP (Session Initiation Protocol) and supports advanced features such as line-rate Ethernet switching, voice encryption, conferencing with four additional callers, and four Ethernet ports.



Source: Zultys Technologies

Jim Barton, a founder of TiVo Inc., pioneered the idea of leveraging open source software in a commercial product, overcoming licensing fears that Linux would not allow proprietary development since it was licensed under the GPL. In an article in *QUEUE* magazine in July 2003, Barton, looking back, wrote that “careful reading of the GPL convinced me that these fears were unfounded and that Linux could give us a powerful development advantage while allowing the protection of our intellectual property.”<sup>22</sup> In order to fulfill the GPL, the company has released the full source code of its modified Linux kernel on its Web site.

Today TiVo is a huge success, igniting a “TiVolution” with some 1.3 million subscribers. TiVo communities have formed on the Internet, like the TiVo Community Forum that boasts roughly 60,000 members and over 1.8 million postings. And unbeknownst to many of TiVo's subscribers, they are using Linux.

Indeed, TiVo appears to the consumer as a “black box,” concealing its open source inner workings. The opposite approach was taken by the Germany-based Dream-Multimedia-TV GmbH for its DVR, called Dreambox DM7000. The company saw an opportunity to target an open source-based product not only at consumers but also at techies – TV device hackers and Linux programmers.



The Dreambox DM7000, a digital video recorder, runs on Linux and is open to all for tinkering. Because of the product's open source approach, much additional software is available for it, including an open source Web server and an e-mail client.

Source: Dream-Multimedia-TV GmbH

"Basically, everybody's free to do anything with this box, but in our standard, tv-centric application we don't use many external applications" says Felix Domke, developer of the DM7000, in an article at LinuxDevices.com.<sup>23</sup> Opening up a product this way would not be possible with proprietary operating systems, which would severely limit outside developers' ability to extend functionality. But thanks to the open platform, much additional software is available for the Dreambox, including an open source Web server and even an e-mail client. Open source software could be the breakthrough needed to finally converge computing and TV.

### ...ON THE ROAD

The growing demand to listen to digital audio files wherever people enjoy music was the key motivation for PhatNoise, founded in 1999, to create a digital jukebox for automobiles. The PhatNoise Car Audio System is an MP3 player that works similar to a CD changer but can hold an enormous amount of music – the equivalent of approximately 400 music CDs. Music is transferred from a PC to the PhatNoise system and played on the car's stereo. The PhatNoise system, which provides intuitive file management and can generate playlists dynamically according to the consumer's preferences, is based on Linux.

Two key reasons for using Linux were \$0 licensing cost and faster development; it was far more cost-effective to start with Linux and adapt it rather than create the

entire solution from scratch. "Embedded devices now need to be equipped with a huge variety of software, and traditional embedded operating systems usually don't have enough built in," explains Dan Benyamin, co-founder and CTO of PhatNoise. "As Linux is often deployed as a desktop operating system, we can start with a much richer pool of applications and features, and pare it down as necessary."

### ...IN YOUR POCKET

There is always room for improvement in being able to access personal information at home or on the road. The availability of a robust operating system with powerful networking capabilities for small devices allows vendors to build cheap servers for accessing personal information.

One is D-Link Systems' Central Home Drive, a modem-sized appliance that boosts the storage capacity of home networks, holding up to 5,000 MP3 files or 20,000 high-resolution JPEG images. Another is Intel's innovative prototype of a personal server, a wallet-sized gadget that links to other computers wirelessly and gives them file access, allowing the user to carry his files and data in his pocket and work anywhere. Lugging a laptop is not necessary.



Linux contributed to innovation and faster time-to-market for the PhatNoise car audio system, which can store hundreds of CDs' worth of music that can be played on the car's stereo system. It was far more cost-effective to start with Linux and adapt it rather than create the entire solution from scratch.

Source: PhatNoise, Inc.

```
int=None, REQUEST=None):
    rendered():
    red(self.pageType
    e().render(self
    ear_cache=0):
    f.clearCache()
    e(*'prerender
    self.pageType()
    t(Permissions.Vt
    REQUEST=None):
    v cached render
    '*)
    v_cooked '*):
    _cooked '*):
    blocks.*)
```





Source: Intel Corporation

Intel's prototype of a personal server is about mobility and convenience. The concept behind the personal server, which runs embedded Linux, is that users carry their data – not their laptop – and work anywhere using local computers. The personal server is like a virtual hard disk that is accessed via a wireless connection to whatever computing device is nearby and available. Users can access their personal information as if working from their own computer, without having to rely on an Internet connection.

personal information management suite, video player, music player and instant messaging. Recently, the company announced the Linux-based Motorola E680™ phone, which is expected to ship globally; this would make it the first Linux-based phone from Motorola available in the United States.

The move to support Linux and Java as core technologies was spurred by innovation and faster development. Motorola's leadership in Java technology, coupled with Linux, gives developers more

In addition to Linux, both appliances use Samba (file sharing server) and HTTP (file sharing protocol), both open source. Intel's Personal Server uses a modified version of Apache to implement the WebDAV service for distributed file management.

freedom to create new applications, from games to productivity tools. Development can be faster than in a proprietary-only world given the rapid pace of the collaborative open source community as it leverages existing open source code.

In the PDA-smart phone arena, Linux is emerging as an alternative to PalmOne's PalmOS, Microsoft's Windows CE and Symbian's Symbian OS. The breakthrough for Linux in the PDA market was the announcement of the Sharp Zaurus SL5000D in 2001; this was the first PDA from a major consumer electronics manufacturer that used Linux as its primary operating system.

Cost was a factor also. A \$0 license fee is very attractive when mass-producing consumer devices; proprietary licensing fees can be a hefty portion of the cost.

With smart phones, the breakthrough came in 2003, when Motorola, the world's number two maker of cellular phones, announced it would base its future cellular phones, including less expensive models, on embedded Linux and Sun's Java programming language. That year, Motorola introduced the world's first open source smart phone, the Motorola A760™, launched in the Asia-Pacific region. The A760 is a high-end smart phone that includes a



Two of Motorola's Linux-based phones, which are like mini multimedia centers, are the A760 (left) and the E680 (below).



Source: Motorola, Inc.



**Even in the laundry:** Open source makes its move into consumer goods as common as laundry products, with “Linux” laundry detergent and “micro&soft” fabric softener, both by Swiss manufacturer Rösch. (According to the company, there is already a “Windows” window cleaning product on the market.) The laundry products are being sold in Wal-Mart in Germany.

Source: Rösch AG

Along with Motorola, Samsung and NEC – two other major device manufacturers – are planning Linux-based phones. Openwave Systems, which provides application and interface software for device manufacturers and cellular phone carriers, and whose software can be found in more than 50 percent of all data-enabled phones shipping today, has also announced a version of its software for Linux.

### ...ONTO THE BATTLEFIELD

Linux may be tucked inside a device in your pocket or living room, but it is also on the front lines for the military. Inside the U.S. Army’s sophisticated wearable computer called the Land Warrior is embedded Linux, powering secure wireless communications and other activities for soldiers using the system.

In general, the wearables and hand-held market is growing, particularly for specialized applications such as those for military, field and other mobile workers. Custom software is an important part of the equation. Adaptability, modularity, power efficiency and time-to-market are all reasons to use embedded Linux in mobile applications.

In developing the Land Warrior system, the U.S. Army migrated the platform from Windows to Linux for

security and other reasons. Working with General Dynamics, the lead contractor for the Land Warrior system, CSC is responsible for delivering the tailored operating system and user application.

### ...AND WHO KNOWS WHERE?

The high profile of open source in the media has led to at least one curious appearance signaling the ultimate in open source consumerization. Wal-Mart’s German branch is marketing a new laundry detergent called “Linux” along with a fabric softener called “micro&soft.” Both products are manufactured by the Swiss company Rösch. Happily, in this combination Linux and micro&soft are highly compatible.

Back in the software world, an open source strategy “under the hood” provides a robust commodity platform for product development, freedom to modify and extend the software, faster time-to-market and lower overall product costs.

A commodity platform raises innovation and differentiation (and profits) to the application level. This is what captures the consumer: the movie, the music, the game, the instant messaging. That open source is “under the hood” is not important to the consumer, nor should it be; the product just works.

```

ent=None, REQUEST
rendered():
red(self.pageType
e().render(self
ear_cache=0):
f.clearCache()
e(*'prerender
self.pageType()
t(Permissions.Vi
REQUEST=None):
v cached render
'')
v_cooked '*):
v_cooked '*):
blocks.*')

```

## MARKET FORCE:

### Open Source Increases Competition, Challenging Established Market Powers

*“Open source is a market force – not just source code.”*  
Forrester Research<sup>24</sup>

Open source is a catalyst for competition in the software market. That competition takes on many forms, involving direct competition with proprietary products, co-opetition, consortia, government initiatives, commoditization and even competition among open source products themselves. The result: open source is stirring the competitive waters, challenging established market powers to defend their products and – if nothing else – serving as a negotiating tool for lowering commercial software license fees.

In the past, competition came from existing companies and start-ups attempting to unseat established market powers. Today the open source community is showing it is a viable source of competition – on a par with, if not stronger than, traditional competitors. As it flexes its muscle, open source is causing all sorts of twists and turns in the competitive landscape.

Even Microsoft has taken exploratory steps into open source. The company has released as open source the code for its Windows Template Library, which is used to develop Windows applications and user interface components, and the code for its Windows Installer XML, which had over 80,000 downloads in the first few weeks after its release. Such moves by Microsoft are notable given the software giant’s business is a paradigm of the closed source approach.

#### HEAD TO HEAD

It is no secret that Linux is going head to head against Microsoft Windows as the operating system of choice, first starting with the server and now moving into the desktop. Linux has put pressure on Windows, which has more than 90 percent of the PC market worldwide (followed by MacOS at roughly two percent). It is a David and Goliath story, to be sure, but David is getting stronger; analysts expect Linux to take over the number two spot in 2004.

An open source alternative to Microsoft has become a rallying cry for Microsoft competitors, for both desktop operating system and the office suite. IBM, HP, Sun, Red Hat, Novell and others have formally backed Linux on the desktop. Novell’s acquisition of SuSE Linux was a bet-the-company move by Novell to face off against Microsoft, a testimony of the strength of Linux as a market force.

OpenOffice, with its roots in Sun’s StarOffice, is competing against Microsoft Office. First released in 2000, OpenOffice has a 14 percent share of the large enterprise office systems market, according to Forrester Research, suggesting it is becoming a true alternative to Microsoft Office, which holds a 94 percent share of the overall office market, has been around for over a decade, and claims 300 million users worldwide. Microsoft, which in general has scoffed the open source approach, nonetheless recognizes the threat; the company published a competitive guide for its sales force comparing Microsoft Office to OpenOffice (which was quickly removed from its Web site). There has even been talk of IBM migrating Microsoft Office to Linux, whether through emulation or actual porting.

The browser wars also sparked a strategic move to open source. In 1998 Netscape released its browser to open source as Mozilla, in a final move against Microsoft’s Internet Explorer. By opening its source code to the worldwide development community, Netscape hoped to garner support for its once-flagship browser. Although the move did little to sustain Netscape, it eventually produced a new open source browser in 2002. Recently the browser, today called Mozilla Firefox, has gained momentum as a competitive browser with innovative features. In an interesting move, Nokia has funded the Mozilla Foundation to develop a cell phone browser. This could fuel Firefox’s popularity and, if it continues to grow, shift the balance of power in the browser world.

## CO-OPETITION

Open source, by its very nature of collaboration, gets competitors to pull in the same direction. Openadaptor is an example of this co-opetition, where competitors work with common code to create an alternative to costly proprietary software. The openadaptor open source project has financial houses Dresdner Kleinwort Wasserstein (DrKW), the investment banking arm of Dresdner Bank, and other German banks on the same side rather than opposing sides. This is notable in the highly competitive banking industry. The banks are working with openadaptor to develop an open source alternative to high-priced Enterprise Application Integration tools.

DrKW developed the original code and released it as open source in 2001. Other banks have adopted and extended the code, in some cases releasing it back to the open source community. And so competitors are working in tandem, if somewhat indirectly, to provide better and cheaper software tools for integrating disparate systems.

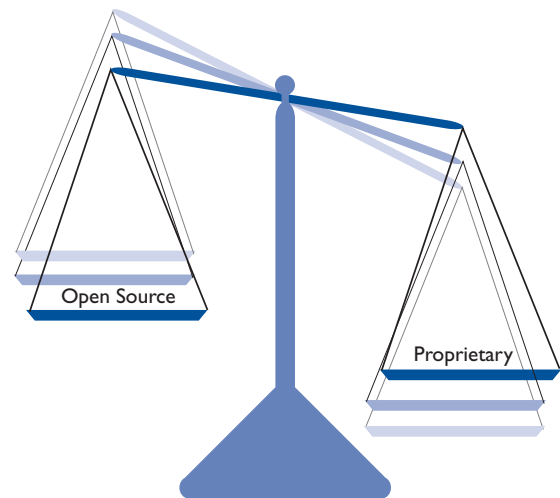
The motivation for DrKW to go open source was to increase innovation by drawing on a larger community, including competitors. Application integration can cost over a million dollars in license fees alone; going open source means no license fees, expertise from a broad community, no vendor lock-in and, ultimately, increased business from making it easier to do business electronically thanks to the integration provided by openadaptor.

Although openadaptor does not offer all the functionality of commercial EAI offerings, it presents an alternative for some organizations and is a catalyst for co-opetition.

## CONSORTIA

The Eclipse consortium, founded by IBM but today an independent consortium, develops products and standards for the Eclipse platform, an integrated development environment used to write applications. As an open source development project, Eclipse has brought together some 50 vendors to create a formidable challenge to established IDE vendors (e.g., Borland Software with its JBuilder IDE).

## OPEN SOURCE SHIFTS THE BALANCE OF POWER



Eclipse was started by IBM as the next version of VisualAge for Java; IBM released the code to open source in 2001 and led the Eclipse consortium for several years, investing \$40 million to seed the open source project. Though IBM's influence over the consortium is questioned – the consortium was just decoupled from IBM this year – the Eclipse platform is a popular alternative that is setting standards for IDEs. The platform receives roughly 30,000 download requests daily and boasts over 600 open source or freeware plug-ins.

Open source, by its very nature of collaboration, gets competitors to pull in the same direction.

But Eclipse is not the only open source IDE. It competes with NetBeans, put into open source by Sun in 2000. (Sun markets the commercial version of NetBeans as Sun ONE Development Studio.)

NetBeans has lost ground to Eclipse, fueling a rivalry between the two. Recently, Sun was asked to join the Eclipse project but declined because it did not want to abandon or merge NetBeans, which is the foundation of Sun's commercial products.



```
port org.apache.  
blic class Stan  
extends Contai  
implements Co  
private trans  
public Stand  
super();  
pipeline.d  
namingRes  
broadcast  
}  
public void a  
synchroniz
```

NetBeans was the first Java IDE that created the very successful development model of having a core system with a set of independent plug-ins. This approach fits the open source distributed development model well and was later adopted by IBM for the Eclipse project.

The Eclipse consortium is a significant player in the IDE environment, showing the power of an open source industry consortium – albeit with a strong industry founder – to set standards and be a force in the market. The Eclipse platform has triggered a new round of competitive products, including IBM's WebSphere Studio Application Developer and SAP's NetWeaver Developer Studio. Both are enhanced versions of the Eclipse platform.

In a move partly aimed at competing with Eclipse and NetBeans, BEA is releasing part of its WebLogic programming framework as open source software. BEA hopes to increase the number of Java developers and encourage them to build products for all Java platforms by providing an open source runtime environment that can deploy applications to any runtime platform, not just BEA's platform. The move is intended to shore up BEA's declining Java application server market position. Anything that makes Java easier to program with is key to growing the Java ecosystem (and countering Microsoft).

## GOVERNMENT INITIATIVES

There have been numerous government initiatives around open source software, from policies mandating its use to policies requiring its review as part of the selection process.

The Massachusetts state government's mandate that open source software must be used in government systems whenever possible, or else it must be explained why open source doesn't meet system needs, has put an entirely new form of pressure on proprietary software vendors. Vendors must justify their products, not to mention their prices, and support open source where they can.

In one case with the state's Health and Human Services Agency, a system was developed on JBoss but then ported to a proprietary environment for

production. JBoss was deemed development-ready but not production-ready by three vendors HHS summoned to review the approach; the vendors raised questions about JBoss performance and how extensively JBoss had been tested. HHS agreed with the vendors' recommendation to not use JBoss in a production environment.

As it turned out, the port to the proprietary platform for production was difficult, eating up any savings realized on the development side. It seems likely that eventually organizations in situations like this will try JBoss for production as well.

Vendors cannot ignore the open source juggernaut but need to live in harmony with it. Look at the open source persistence framework Hibernate; whereas JBoss integrates well with Hibernate, other vendor platforms do not. Instead, they require custom code. This was one element of the porting difficulties HHS faced.

Government initiatives promoting open source compel vendors to explain what is uniquely different about their products, and why an organization needs their products' particular features. Thus, software assessments are becoming more granular, based on specific features currently needed; the sales pitch that features may be needed at a later date does not seem to hold as much sway as it once did. Vendors will need to seek new ways to compete as they face off against open source offerings and mandates.

## COMMODITIZATION

Open source software brings commoditization to the software stack, starting at the lowest levels. In theory, this drives out competition over time (the way the open hardware platform of the IBM PC drove out proprietary hardware vendors Data General, Digital and Prime). The open source software products become commodities, creating a low- to no-price market where proprietary products cannot compete.

This can be seen with three open source utilities: Log4J, Tomcat and Struts. These three are used in the majority of enterprise applications today and have no real commercial competition. They have taken over their market space; prior to these utilities, developers had to



create custom code to do the job. (Log4J is for logging services, Tomcat is the Java “container” or processor for Java Servlets and Java Server Pages, and Struts is a framework for building Java Web applications.) All three are part of the Apache Jakarta Project, an open source project for commoditizing enterprise Java application development.

As commodity software, Log4J, Tomcat and Struts preempt would-be competitors from entering the market. Users get the software they need; the open source products are so good and widely used that they don’t attract commercial competitors.

With commodity software as a base, differentiation moves up the stack. The Eclipse platform is essentially a commodity platform, upon which vendors build products. CSC has created one such product, VP/MS Model Manager, an IDE customized for the financial services industry to develop new financial and insurance products. VP/MS Model Manager is part of VP/MS, CSC’s Visual Product Modeling System. VP/MS has been integrated into many of CSC’s financial and insurance services products.

Using VP/MS and VP/MS Model Manager, insurance companies like Signal Iduna in Germany are able to bring new products from concept to market in as little as half the time and half the effort it used to take.

By working with a commodity open source platform for the model manager, CSC was able to focus on custom enhancements rather than reinventing the wheel. “The customer gets a higher quality product than if open source had not been used,” says Rolf Wilms, senior software architect in CSC’s Financial Services Group in Cologne, Germany. “The extensible plug-in architecture offered by Eclipse allows us to incrementally add support for the VP/MS family of products.”

With VP/MS Model Manager, CSC did not just take from the open source community but also gave back. CSC contributed two modules it developed back to the Eclipse community: Module Management (for configuration management) and Editors (for detecting conflicts before they occur).

“We could not have developed VP/MS Model Manager without Eclipse. We would have had to develop the entire IDE first, which would have been too costly. It’s only fair that we gave something back to the community,” Wilms explains. “We contributed something of technical value to the community that was not a competitive advantage for CSC.”

And so a commodity IDE stoked innovation, both for CSC clients and for the open source community at large.

This is good news for the market, but it must be noted that commoditization – and its competitive impact – does not happen over night. For instance, commoditization threatens the database arena but hasn’t taken hold yet. MySQL is making inroads, growing over 30 percent in the last half of 2003, and MaxDB was a valiant effort by SAP, in conjunction with MySQL, to release a database management system into open source. The presence of open source databases signals the evolution of the database to commodity software.

However, an open source database is a tough sell at the high end, where most enterprises will continue to use proprietary databases (Oracle and IBM’s DB2) for some time. Companies have their businesses invested in these databases and do not consider them commodity software just yet. Switching costs are high in terms of data migration, and many companies enjoy favorable database license terms, negating the effect of free open source licenses. Thus, while the database arena shows movement towards commoditization, open source adoption will be slower for high-end databases than for other software like operating systems and development tools.

Open source databases are gaining traction at the edge of the enterprise for tasks like reporting, where the technology can just be “good enough.” As more open source databases are deployed at the edge and the low end, and confidence is built, the stage is set for open source databases to move into the core of the business. Watch how this space plays out.

```
int=None, REQUEST=None):
    rendered():
    red(self.pageType
    e().render(self
    ear_cache=0):
    f.clearCache()
    e(*'prerender
    self.pageType()
    t(Permissions.Vt
    REQUEST=None):
    v cached render
    '*)
    vcooked'*)
    cooked'*)
    blocks'*)
```

## STANDARDS

Standards change the nature of competition. They represent requirements vendors must conform to in order to attract the largest possible market (although critics will argue that standards inhibit innovation.) Value is added on top of the standard. Once again, as with commoditization, differentiation moves up the stack.

The open source community can accelerate the development of standards. As the world gets faster, the open source community, in its global networked collaboration, has the potential to work through standards issues faster than traditional standards bodies, which can get bogged down in face-to-face meetings and bureaucracy.

One example of an open source community creating a standard is Hibernate. The Hibernate project was started in late 2001 by a student in Australia, Gavin King, and is now the de facto standard for object-relational mapping tools in small and medium applications. Another standard, JDO (Java Data Objects), was under development at the time but has floundered in the face of Hibernate.

Open source is a competitive force, redistributing the balance of power in the open source world and, more significantly, the software market as a whole.

Hibernate is a good, lightweight tool. It is straightforward, easy to use, and has good performance. It was not a standard when it was initially developed, but because so many people started using it, it became the de facto standard for small and mid-size organizations.

JDO, by comparison, is complex to use. Although major vendors have created products based on JDO, they are not widely used and in some cases the product only partially implements the standard. Many people were involved in creating JDO, which took three

years to reach its first release, in 2002. At the time the standard was incomplete, and it has never been finished. So while Hibernate has thrived, JDO has struggled. However, there are signs that the forthcoming JDO 2.0 is gaining ground, and Oracle's TopLink and Hibernate are considering supporting it. In fact, Gavin King is now part of the JDO standards body, which he was asked to join.

Recently the EJB3.0 Expert Group, which is shaping the next version of the Enterprise Java Beans (EJB) specification (EJB constitutes a major part of the J2EE specification), decided to move its current persistency model to a new model based on Hibernate's concepts rather than those of JDO. This illustrates how the agile and pragmatic approach of open source projects can be superior to the institutional definition of standards.

Is the open source community the new route to standards in the 21st century? Time will tell. IBM is pressing Sun for an open source Java so that an independent community can further develop the technology that has become an industry standard. At the very least, the open source community is an alternative approach to creating standards, and the competitive ramifications are being felt.

## SURVIVAL

Another force at work in the open source competitive landscape is sheer survival.

Sometimes the hype about open source software gives the impression that all open source software is good, which is *not* the case. As with commercial software, only a select few products are really good and widely used. There are over 82,000 open source projects on SourceForge, but probably 80-90 percent of them are low-quality or low-usage projects.

Open source projects survive based on their quality and functionality, not marketing hype. The best products survive, such as Linux, but so do products that are "good enough," such as MySQL.

Within the open source world, survival is not necessarily a dog-eat-dog proposition either. One of the

biggest open source rivalries is between desktop environments KDE and GNOME, and the two look like they will co-exist.

Open source is a competitive force, redistributing the balance of power in the open source world and, more significantly, the software market as a whole. The open source community is perhaps a surprising competitor, being a volunteer corps, but the quality

and passion behind the movement are standing up to major market players. As a 2003 study by the Swedish Agency for Public Management concluded, “Free and open source software is not any makeshift phenomenon, but instead a fully adequate and dependable competitor to existing proprietary products and solutions.”<sup>25</sup> Open source is a competitive force that organizations, whether software consumers or producers, need to recognize and leverage.

## NEW DOMAINS:

### Open Source Lives in Many Domains

It is an age-old longing: having access to all the brainpower of mankind. In 1937 British writer H.G. Wells wrote an article about a “permanent world encyclopedia” in which he predicted, “A great number of workers would be engaged perpetually in perfecting this index of human knowledge and keeping it up to date.”<sup>26</sup> Wells’ vision was prophetic, as this is exactly what Wikipedia is about today.

Wikipedia is a global open source encyclopedia on the Web to which anyone can contribute. Billed as “the free encyclopedia,” Wikipedia began in 2001 and today boasts over 230,000 articles in the English language version. The open source approach enables Wikipedia to have the most accurate, up-to-date information possible. Anyone can contribute information and anyone can access the information.

Wikipedia, which received the Prix Ars Electronica top prize for digital communities in 2004, is an example of how the open source approach lives in many domains, not just software development, and brings a wider meaning to “source.”

The general characteristics of the open source movement, wherever it operates, are 1) freely available information and 2) collaborative problem-solving. Usually the problem is complex and communal in nature. Often the open source approach is a way to

match demand (a problem needs to be solved) with supply (a volunteer community of experts).

Of course, communities have been solving problems for a long time, whether to fight disease, invent a product or catch a criminal. What makes today’s open source movement different is the Internet, which links



**WIKIPEDIA**  
*The Free Encyclopedia*

Open source concepts are at the core of Wikipedia, the online encyclopedia that is freely available and modifiable. ([www.wikipedia.org](http://www.wikipedia.org))

```
port org.apache.  
blic class Stand  
extends Contai  
implements Cor  
private trans  
public Stand  
super();  
pipeline;  
namingRes  
broadcast  
}  
public void a  
synchroniz
```

the community with astonishing ease, speed and reach, and provides access to core information related to the problem.

The Human Genome Project is an example of a well-known initiative that represents open source in another domain, biology. Researchers around the world work together to decode human DNA, and the results are freely available to all. A less well-known initiative is Openlaw, an experiment at Harvard Law School for creating legal arguments in an open source-style public forum, enabling editing and commenting on legal briefs online. The merits of open source are being put to work in these and other areas far outside the realm of software development, including hardware and product design, agriculture, science, music and collaborative knowledge.

#### HARDWARE AND PRODUCT DESIGN

ThinkCycle is a collaborative initiative for designing products and services for under-served communities and the environment. Founded on the open source approach, ThinkCycle has created a global community to share knowledge and develop the latest innovations for addressing critical problems in such areas as health, energy, education and sustainable living (agriculture, water, housing). Projects include low-cost eye glasses, low-cost cholera treatment devices, human power generation (in lieu of batteries), special incubators for premature babies, and household water treatment systems.

The ThinkCycle initiative, developed and operated by the MIT Media Lab, joins designers, engineers and experts from universities, non-governmental organizations, companies, local communities and the general public. The idea is to provide a database of well-formed problems and solutions that these groups can contribute to and access.

In the computer market, OpenCores.org is a portal for open source hardware projects, somewhat akin to SourceForge for software projects. OpenCores focuses on projects with openly developed application-specific integrated circuits (ASICs). Considerable effort goes into ASIC design; re-using core designs speeds up ASIC design by enabling engineers to collaborate.

The mission of OpenCores is to help the hardware community create and publish core designs under a license for hardware modeled on the Lesser General Public License for software. (See Legal and Business Issues.) The intent is to have core designs that are freely available, freely usable and freely re-usable.

The benefit to the market is innovation; anything that spurs creativity – like reuse of core designs – is key. IBM is beginning to open up its chip design, a significant move for a company renown for its advanced chip design and manufacturing technology. But in a field where future innovation is expected to come more from customization than increased speed, IBM hopes to seed that innovation by opening some of its chip technology to a wider community.

Following lessons it has learned from the Internet and Linux, IBM intends to share more technical information about its Power family of microprocessors, distribute free software tools for chip design and testing, and establish design centers worldwide to help customers create custom chips.

Intel as well is looking at open source for advancing its chip design. Intel's Open Source Machine Learning Library (OpenML) enables Intel to gather input from a large community to learn more about the nascent field of real-time machine learning. The more Intel understands about machine learning and how it is being used by others, the better it can develop chips that facilitate it. Open source expands the research base and stimulates collaboration and innovation in machine learning, ultimately leading to improved chip design.

The open-source-for-innovation theme carries through even to large-scale buildings. MIT's Open Source Building Alliance focuses on developing and testing new strategies that will lead to better materials, technologies, applications and services for buildings. Initially, the project is concentrating on single and multi-family residences.

There is a conscious effort to adapt open source software techniques to the building domain. According to the OSBA prospectus:

Open source strategies have dramatically improved the quality, value, and variety of products in other industries, from electronics to automotive. In contrast, most new homes and apartments are generic, low-grade, and expensive. The OSBA proposes that an open source web of industrial relationships, combined with the modularity of design, data, electronics, software, and physical component connections, can lead to an explosion of creative activity resulting in high-performance, cost-effective environments. We believe that this approach is necessary to remove barriers to innovation, and that it will create exciting opportunities related to energy conservation, proactive home-based preventative health care, and new forms of work, learning, entertainment, and the mass-customization of highly personalized residential environments.<sup>27</sup>

OSBA will operate as an open source organization, with a Web site for idea generation, technical evaluation and public comment. The organization will create test beds and demonstration environments, some of which will be rendered in commercial developments.

Some of the design concepts under consideration are modular building parts, reconfigurable components that replace walls, power connections, and technologies for environmental sensing, activity recognition and communication media.

## AGRICULTURE

In addition to shelter, open source techniques can be applied to another basic need: food. Cambia (Center for the Application of Molecular Biology to International Agriculture) conducts research in sustainable agriculture, developing a powerful genetics and policy toolkit that enables plant breeders and scientists worldwide to add new directions to conventional plant and animal breeding. Cambia's goal is to help local communities solve their own agricultural and environmental problems by unlocking the intellectual property logjam that has stymied the full potential of modern genetics.

Cambia, based in Canberra, Australia, believes everyone from farmers to researchers should have access to the latest biotechnology tools; they should not be controlled

“Preprint servers” emerged in the early 1990s as a way to provide fast access, via free downloads, to articles that have been peer-reviewed and submitted to journals but have not been published yet (hence the term “preprint”). In the late 1990s, taking the next step, the preprint movement spawned the Open Archives Initiative, which looked at how independent preprint server efforts could cooperate to expand the openness of scholarly publishing in general. The Open Archives Initiative develops and promotes interoperability standards for



efficient content distribution. Although the initiative has its roots in scientific publishing, it is developing standards for open access to a wide range of information, including social sciences and the humanities.

Another initiative tackling the challenge of timely, accessible scientific materials is the Public Library of Science. PLoS is a nonprofit organization of scientists and physicians committed to making the world's scientific and medical literature a public resource – that is, available free of charge to anyone anywhere. “Immediate unrestricted access to scientific ideas, methods, results, and conclusions will speed the progress of science and medicine,” the PLoS Web site states.<sup>31</sup>

To this end, PLoS has two journals, PLoS Biology and PLoS Medicine, in print and online (PLOS Medicine launches in fall 2004). The journals contain full text and data of published research articles. PLoS was awarded the Wired Rave Award in Science in 2004 by *Wired* magazine, which lauded PLoS for “cracking the spine of the science cartel” with its model for open-access scientific publishing.<sup>32</sup>

Some have also called for an open source approach to invent drugs that fight tropical diseases. A Web site would be used by biologists and chemists to volunteer their expertise in specific areas of certain diseases. Scientists would work collaboratively, annotating databases, conducting experiments and sharing results openly, including having chat room discussions about results. Ultimately, drug development would be awarded to a laboratory via competitive bid, but the drug would be released in the public domain for generic manufacturers to produce, thus reaching the widest audience at the lowest price.

## MUSIC

Whereas open source in publishing refers mostly to freely available information, in the world of music we get back to the dual core characteristics of open source: freely available information *and* collaborative creation. The Open Music Registry promotes open source music – music that is freely shared, listened to, modified and distributed. It is a powerful tool for musicians and those who love music. Though the site is dedicated to non-commercial music, it is a proponent of artist compensation and ownership (copyright) of musical works.

Music at the site is licensed under the Open Audio License, created by the Electronic Frontier Foundation. The license, founded on open source software principles, states that its goal is to restore music copyright's original purpose: “the creation of a vibrant public commons of music that we all can enjoy and that artists can build upon.” (Although the Open Audio License was the driving force behind the Open Music Registry, launched in 2001, today there is a small selection of similar licenses artists can use instead of the Open Audio License.) The registry is catching on; from 2003 through March 2004 there have been over 56,000 downloads of free music, or an average of over 120 downloads a day.

Certainly music lends itself to open source, being digital in form (and thus malleable and shareable like software) and having already tested the waters of freedom with peer-to-peer file sharing. Where music goes with open source will be an interesting test of the creative-commercial balance.

Meanwhile, if music is open source, what about the air waves that carry it? The idea of open source radio spectrum is being championed by David Reed, an inventor of TCP/IP and creator of Reed's Law of group-forming networks. (See the LEF report “The Architecture rEvolution.”<sup>33</sup>) Reed wants to open up a portion of the radio spectrum and protect it via the Creative Commons organization, a nonprofit dedicated to creating public goods while preserving some private rights.

Reed argues that spectrum is *not* a scarce resource but is plentiful and should be free to anyone to use. Users should be organized into a cooperative network, not as discrete channels. By deploying a software-defined radio device, each user added to the network increases rather than decreases network capacity – just like adding a node to the Internet increases its capacity. Innovation occurs at the edge of the network, where the users are; anyone with a good idea can broadcast it without requiring permission from a central authority or owner. In Reed's vision, open spectrum brings innovation in service of the public good.

## COLLABORATIVE KNOWLEDGE

The notion of harvesting knowledge via open source methods is well-exemplified by Wikipedia. In addition to an open content encyclopedia, other examples of open source collaborative knowledge are appearing in text books, university courses and even cookbooks.

The Open Textbook Project is about creating text books using open source methods. The books are to be developed collaboratively and made available broadly in the United States and elsewhere, ideally reaching underserved markets. The collaborative model helps ensure that material is up-to-date and tailored for local audiences. Another project is Wikibooks, which creates and disseminates open content text books, manuals and other texts and currently has over 80 books in development.

In a similar spirit of access to knowledge regardless of geography or income, MIT's OpenCourseWare initiative aims to distribute the university's course material free of charge via the Web to learners far and wide. The idea for the initiative, formally launched in 2003, came from the open source development model. With material from 700 courses currently available,

and plans to incorporate all classes by 2008, MIT hopes to revolutionize the way universities share information and people learn.

On a lighter note, if open source can make you a smarter person, it can also make you a better cook. Several open source cookbooks are gracing the shelves of cyberspace, inviting cooks of all kinds to use and submit recipes. These constantly evolving works include *The Open Source Cookbook: Fuel for Geeks*, *International Recipes Online*, the *Cookbook of the Unix Group at the University of Kaiserslautern* (recipes in German), and IT publisher O'Reilly & Associates' *Community Cookbook*, which plays off of O'Reilly's programming cookbook series (e.g., *Apache Cookbook*).

From cooking to crops to cholera treatment, open source is sparking innovation in numerous domains. With this comes an ideology of openness: providing global access, enhancing knowledge and solving problems for the greater good. As the Internet spills over into our personal lives and all areas of endeavor, more people outside the software arena will become aware of open source and see its potential in many new domains.

## FUN FACTOR: Open Source Gets Your Creative Juices Flowing

Open source has been likened to applied science: developers, like scientists, explore new ideas that are interesting and fun. They can't wait to get their hands on the code (or get back to the lab) to experiment and tinker. They are driven by curiosity, a desire to learn and share their findings with others, and a desire to contribute something meaningful to the community.

For developers (or scientists), it is fun to solve a problem, and they take personal pride in it. Because it's fun, they spend hours programming highly original or useful tools, with amazing creativity and dedication.

By the same token, these creative minds will stop programming if they perceive it is no longer fun. Organizations need to recognize that the fun factor is a palpable element of open source development and nurture it for project success. Don't confuse "having fun" with "having a party;" professional enthusiasm and enjoyment contribute to individual performance. Motivated employees who have fun on the job tend to be more productive than others.

## IT'S ABOUT ATTITUDE, NOT AGE

There are people who develop for results and people who develop for fame. Solving challenging problems is fun. Creating a useful or fancy piece of software and

being able to use it within minutes – no laborious production process needed – is fun. Being known and respected by colleagues, and even users, is fun.

It is also fun to “beat the system” and develop code free from capitalistic constraints. (See Software Revolution.) It’s cool to develop open source software. If you want to prove that you’re smarter than the big corporations, and you don’t want to do anything illegal, you do open source development. This is not a question of age but attitude.

Of course, fun is subjective; what’s fun for one person may be dreary and tedious for another. That is where the community comes in, as a hedge for varying developer interests. Someone in the community will find a particular feature or problem fun to work on, and it will get done – though perhaps not on a strict schedule, which can be a problem in open source development. (More on that later.)

## FUN AND GAMES

Open source development gives developers an opportunity to show off their talents. They can work on what interests them, independent of work or school. And so you see open source applications for a variety of hobbies: a route-finder for cyclists ([BBBike.de](http://BBBike.de)), a beer site for home brewing ([CyberBrau.org](http://CyberBrau.org)), an ebook manager ([eBookCollector.sourceforge.net](http://eBookCollector.sourceforge.net)), a family genealogy tree ([vjet.demon.co.uk/ftree](http://vjet.demon.co.uk/ftree)).

Open source development gives developers an opportunity to show off their talents.

Larger-scale projects focus on hobbies almost everyone is interested in, like music or film. The Ogg multimedia project, with its Ogg Vorbis open source audio format (à la MP3), and the VideoLAN project, an open source client-server video streaming solution, both offer high-quality compression techniques for streaming audio or video with no licensing fees.

Also in the video world, DivXNetworks worked on an open source video codec (compression/decompression

software needed to play video files) as part of Project Mayo, home of the open source video codec project. Although the product was not released as open source software – flaming controversy that the company used the open source model to tap others’ ideas – a different product stemming from Project Mayo, XviD, ultimately was released as open source. XviD is a high-quality MPEG-4 codec, with excellent compressibility and advanced features. Because it is open source it is free of license charges, customizable and has good cross-compatibility among desktop operating systems.

Music and video are great fun as open source projects, to be sure, but perhaps the ultimate in fun software is computer games. Interestingly, most open source games have not been particularly state-of-the-art, dogged by poor screen design and low-quality animation. However, this is changing as recent entrants have been increasingly sophisticated.

One of these is PlaneShift, a multiplayer virtual fantasy world with state-of-the-art 3D graphics. PlaneShift is very active, with 20 core developers in the United States, Europe, Canada and New Zealand and several sub-projects. The fact that it is open source means that users, as developers, can create elements like their own house or castle, and everyone who plays can use these creations and build on them to make new ones. As PlaneShift’s Web site says, because of open source development “the game will be expanded endlessly for years to come.”<sup>34</sup>

PlaneShift believes strongly in open source, and in successful and enjoyable games. For this reason the company employs three licenses. It uses the GPL for all source code except the rules (logic) of how the game is played, and two more proprietary licenses for the non-code assets of the game, such as artwork and text. In justifying this strategy, the PlaneShift Web site states:

We have seen too many [game] projects fail, and it’s painful to see more when we know how much effort has been put into them. We want to find a way to gather all these individualistic developers and create a successful project for the players, but also for those developers that share the same dream.<sup>35</sup>





PlaneShift, a 3D fantasy game, is one of the most sophisticated open source multiplayer games. ([www.planeshift.it](http://www.planeshift.it))

Source: PlaneShift

PlaneShift uses a mixed licensing scheme to support two main goals: to keep resources united under one project and avoid forked projects, and to ensure success through uniqueness of the game. PlaneShift's licenses facilitate creativity among the development community, while preserving as unique the creativity of artists and designers who contribute.

A less graphically sophisticated, though no less challenging, open source game is NetHack, a single player dungeon exploration game. Despite the game's low-tech ASCII interface, it is feature-rich in terms of its logic, special levels, characters, items and concepts. NetHack, first released in the 1980s, has a strong, almost cult-like

following and has even been featured in a comic strip. The NetHack community provides a variety of play scenarios and interfaces that are a constant source of fun and exploration for this Dungeons and Dragons-inspired game.

## SERIOUS BUSINESS

Games are fun – a playground for the imagination and for attempting new technical feats. But games are also serious business. For some game and toy companies, open source is figuring into the business strategy, with positive results.

Consider Doom, one of the grandfathers of computer games. When

Doom was released in 1993, it had ground-breaking technology and superior playability. Keeping the source code of the game's engine, or platform, proprietary was an economically sound choice for the creator, id Software.

But as similar engines emerged, the value of the Doom source code diminished. In 1997, id Software released the Doom source code as open source under the GPL. The company had a new engine about to come out, so it decided to give away the current engine and position the company as supporting open source. Other game engines, notably id Software's Quake, were later released under the GPL also.



BioWare's multiplayer game Neverwinter Nights comes with an editing tool that allows players to create and share adventures of their own. This open strategy was inspired by the open source movement. The screen shot shows a player's character negotiating with some NPCs (non-player characters) in the environment originally created by BioWare.

Source: Neverwinter Nights screenshot courtesy of Atari Interactive, Inc.(publisher). © 2002 Atari Interactive, Inc. All rights reserved. Used with permission.



```
port org.apache.  
blic class Stan  
extends Contai  
implements Co  
private trans  
public Stand  
super();  
pipeline;  
namingRes  
broadcast  
public void a  
synchroniz
```

BioWare, creator of *Neverwinter Nights*, also follows an open strategy but in a different way. Instead of laying open its source code, BioWare sells a toolset that enables users to create entire worlds and characters of their own and share them with others. This was a revolutionary idea at the time of the game's release, 2002, and was inspired by open source.

"We had been watching the open source movement with a sharp eye and we saw the emerging strength of open collaboration," recalls Trent Oster, producer of *Neverwinter Nights*. "When we first started talking about the editing tool we were concerned that creating a module on your own would be too much work. Our hope was that, in an open source fashion, the fans of the game would begin to share work and expertise. Soon after the game shipped we saw the strength of our community as they divided work among experts and quickly began supporting the development efforts of various groups. To date the community has accomplished some truly amazing things through this collaboration."

*Neverwinter Nights* has been a great success, with 1.7 million registered users and 30,000 unique users logging on per day. There are over 2,700 self-created modules available at no charge via the Internet. These creations are extending the lifespan of the game, which continues to sell well today. *Neverwinter Nights* earned a spot in *FiringSquad's* Top 10 PC Games list in 2004.

Another place where an open approach has become part of the business strategy is at toy maker LEGO. The company's LEGO MINDSTORMS programmable robotic toys, introduced in 1998, were reverse-engineered years ago by well-intentioned hackers. As schematics were posted on the Internet and open source development tools began to appear, the company decided to embrace the

open movement, opening up the product specifications and providing support for open source programmers. LEGO made the product easier to program and published advanced programmer kits and documentation on the Internet, at no charge.

The results have been impressive. According to one account in *Software Development Times*: "A bustling community has grown up around Mindstorms. Open source development systems based on C, Java and Forth have appeared. Programmers share expertise and source code on dozens of news servers and Web-based community sites, including sample code in C++, Perl, Tcl, Visual Basic, Delphi and Scheme. One enterprising group of programmers has even developed an open-source operating system for the robots, allowing developers to control the devices in arbitrary and complex ways."<sup>36</sup>

LEGO's move into open source dovetails nicely with its slogan "Play On." The slogan, as described in a company profile, "focuses on open play, which can continue



LEGO MINDSTORMS robots have enjoyed attention from the open source community. LEGO has supported open source programmers eager to dig into the inner workings of the programmable robotic toys and create new features.

Source: The LEGO Group

Play inspires the worlds of government and business too. Connections between computer games and the military go deep, for training and simulation exercises. Scenario planning in business, as well as modeling and simulation, draw from gaming. Doom has even been considered for a system administration tool, as an interface that provides a more intuitive environment. (Kill that errant process!)

So, while games are about fun and play on the surface, they go deeper than that. Organizations can apply lessons from open source games to their IT strategy, integrating creativity with efficient project management. Organizations can encourage the open source community to focus on extensions that differentiate an existing product or platform, thereby extending its lifecycle. The organization is the hub of core development and provides the tools the community needs to be creative.

Also, organizations are always desperate for IT talent. Smart ones will scout the game world, both open and closed source, for creative technical minds.

```

getFactory().getLog(StandardContext.class);
}

@Override
public StandardContextValve() {
    super(this);
    ApplicationBroadcasterSupport();
}

@Override
public void setParameter(ApplicationParameter parameter) {
    if (parameter != null) {
        parameters.add(parameter);
    }
}

@Override
public List<ApplicationParameter> getParameters() {
    return parameters;
}

@Override
public String getName() {
    return "applicationParameters";
}

@Override
public int getLength() {
    return parameters.length;
}

@Override
public ApplicationParameter getParameter(int index) {
    return parameters.get(index);
}

@Override
public boolean hasOverride() {
    return false;
}

@Override
public void clearCache() {
    // security.declareProtected(Permissions.VIEW_RUNTIME, this::clearCache);
    // *def* *clearCache*(self, REQUEST=None):
    //     """
    //     forcibly clear out any cached render
    //     """
    //     self.setPreRendered('*')
    //     *if* hasattr(self, '_v_cooked'):
    //         delattr(self, '_v_cooked')
    //         delattr(self, '_v_blocks')
}

```

# LEGAL AND BUSINESS ISSUES\*

Open source legal and business issues need to be taken seriously. Intellectual property issues are at stake, unlike in a closed source world where source code is not shared. Organizations need to be disciplined about their use of open source software. Adhering to software licensing rules is not new, but what is new is the wide array of license obligations or requirements introduced by the open source movement. Organizations must consider a number of legal, commercial and policy issues as they embark on using open source software.

## LEGAL CONSIDERATIONS

The Treasure Chest shows there are endless opportunities for realizing business value from open source software. However, before an organization uses open source software, it must understand that although the source code may be free to use, it is not free of obligations. Non-compliance with these obligations could negate the organization's right to use the software, and in certain circumstances non-compliance could signal breach of contract, making the organization potentially liable for financial damages.

There are two key points to understand about using open source software:

- Unless the software has been put into the public domain (that is, the author has given up to the general public all rights of ownership, including copyright over his or her created work), one's access to open source software is subject to stated conditions of use determined by the owner. These conditions are the license terms.
- The license terms may be different than other more familiar software license terms, and in some cases the open source license terms may defeat the reason you wanted to use open source software in the first place.

If we start with the premise that Linux is the best known example of open source software, then it is important to understand that Linux was made available to the community by Linus Torvalds using a license agreement created by Richard Stallman, founder of the Free Software Foundation. That license is the General Public License.

## The GPL – Free Speech, Not Free Beer

The GPL is more than a set of terms and conditions of use. It is also a political manifesto that sets out the vision for a widespread software community where software is developed by a collaborative effort and made freely available to all.

This set the early benchmark for the trade-off for the right to use open source software.

Stallman refers to four types of freedoms that enshrine free software:

1. the freedom to run the program for any purpose
2. the freedom to study how the program works and to adapt it to your needs (access to the source code is a precondition for this)
3. the freedom to redistribute copies so you can help your neighbor

\* This chapter raises legal issues that may be relevant and should be considered in certain circumstances when open source software is to be used. This chapter is not intended to be exhaustive or comprehensive, nor does it constitute legal advice on which organizations should rely. If in doubt, consult your legal advisor.

Legal commentators representing the various interests of open source advocates and who were among the first to identify or comment on legal issues surrounding the use of open source software include Eben Moglen (Professor of Law & Legal History at Columbia Law School and General Counsel for the Free Software Foundation), Lawrence Rosen (attorney and founding partner of Rosenlaw & Einschlag and past General Counsel and Secretary for the Open Source Initiative) and Lawrence Lessig (Professor of Law at Stanford University).

4. the freedom to improve the program and release your improvements to the public, so that the whole community benefits (again, access to the source code is a precondition for this).

Stallman has famously described these freedoms, and the concept of free software, as being free in the sense of free speech, not free beer. It's about liberty, not price.

### The Catch

Although these four freedoms are incorporated in the GPL, the GPL also includes a number of restrictions. In particular, the GPL has a provision that any new work that contains, in whole or in part, open source software licensed under the GPL must also be licensed under the GPL.

Therefore, a software developer who combines software licensed under the GPL with his or her own proprietary software, thereby creating a derivative work for distribution or sale to a third party, would be required to license his or her own proprietary software under the GPL. This result has been described as “viral” or “infectious” in nature because the GPL terms self-replicate to the original program. (Supporters of the GPL would say that the impact is not viral at all but based on reciprocity.) Thus, the software developer would be required to publish his or her entire source code. This applies even if only a few lines of GPL code are incorporated in the developer's proprietary software.

This has unnerved software development organizations that have expended many millions of R&D dollars to create source code that is viewed as proprietary and therefore closed code.

For example, a program that incorporates both open source and proprietary code in a single code block would require the proprietary source code to be disclosed to ensure compliance with the terms of the GPL.

This is the basis of the concept of “copyleft.” A copylefted license reverses the traditional licensing concepts by using the license to give the licensees more freedom rather than more restrictions (which is usually the case where the copyright owner has the exclusive rights). So copyleft mandates that everyone

has the right to use, modify and redistribute a program's code or any program derived from it upon the same terms. (Now, an attorney may argue that this is just a different use of copyright law; instead of stating that there is a requirement to pay \$X for a license to use, the requirement of the owner in return for the right to use is to provide a copy of the subsequent developer's source code.)

If you think this is all theoretical or academic, think again. Less than a week after Sun pledged to make available the source code of its Solaris operating system to the open source community, SCO was reported in the press as stating that its license restrictions prevent Sun from contributing Solaris to the GPL because Solaris includes code licensed from SCO under another form of license. (The *SCO v IBM* case is discussed later.)

If Sun did release the Solaris open source code under the GPL, then theoretically it would also have to release those parts of the Solaris code that have been developed in-house.

It should be emphasized that this viral effect applies only to the creation of a derivative work based on a GPL-licensed work, as distinct from running a proprietary work *with* a GPL-licensed work. In the latter case, using Linux as an example of a GPL-licensed work, a proprietary program running with Linux neither contains nor is derived from Linux, so the viral effect does not apply.

### The Concession

Recognizing the potency of the viral effect, the Free Software Foundation subsequently introduced the GNU Lesser General Public License (LGPL), which permits LGPL-licensed software libraries to be linked to non-LGPL-licensed software in a less restrictive way. The LGPL allows proprietary software to be used in connection with GPL software through the use of programming libraries without requiring the proprietary software to be subject to the GPL (or LGPL) provisions. This concession was made as “there may be a special need to encourage the widest possible use of a certain library so that it becomes a de-facto standard.”<sup>37</sup>

```
nParameters.length, results, 0, length)); parameter; ", parameter));  
"ropper"));
```



Two examples illustrate how the LGPL works:

- An LGPL library “X” can be linked with BSD-style licensed software “Z” and the resulting package “XZ” can be distributed as a whole without requiring that “Z” be licensed under the LGPL.
- OpenOffice.org, which makes its source code available under the GPL, licenses the libraries and component functionality of its Opensource.org source code under the LGPL. OpenOffice.org also supports the dual-license model by making a version of the software available commercially using the Sun Industry Standards Source License. (The dual-license model is discussed later.)

The above analysis is intended to draw attention to a concern with a particular requirement of the GPL. For a software author who wishes his or her code to be freely available, and who hopes to receive contributions from other users of his code, the GPL may be ideal. This is particularly the case if the author holds the view that no one should profit from the author’s work by seeking to exploit the fruits of another person’s labor.

But is forcing someone to release their source code consistent with the concept of free software? As the author, you may want other users to do as they wish with your code. However, perhaps by shutting out the software development organization that historically has distributed closed source code, the user community misses out on positive contributions – or the organization develops its own (non-infringing) version.

In addition, if a traditional closed source organization were to incorporate open source code into its products, it may actually benefit the open source community because it may be easier for customers of that organization to subsequently switch from the proprietary product to the open source equivalent. There may be more job opportunities for open source software developers inside the organization once its products incorporate open source code. Some food for thought.

### Criteria – A Plethora of Licenses

Because of the restrictive nature of the GPL and the connotations of the word “free” when applied to software championed by the Free Software Foundation, the Open Source Initiative was created in the

late 1990s and the term “open source” was officially adopted.

Two expressions often used interchangeably are “free software” and “open source software.” Although the distinction may be academic to many, to the supporters of the Free Software Foundation on the one hand, and the Open Source Initiative on the other, the distinction is the subject of heated debate – a purist versus pragmatic view. This chapter employs the more frequently used “open source software” unless otherwise required.

The leaders of the Open Source Initiative (including open source advocate Eric Raymond) created the Open Source Definition, a set of standard criteria that open source licenses are required to incorporate so as to be “OSI Certified.” These criteria include:

- **Free redistribution** – The user has an unlimited right to give away the software royalty-free to third parties.
- **Source code** – Source code must be available in a form suitable for modification.
- **Derivative works** – Derivative works and other modifications must be permissible and must allow for their distribution under the same terms as the license of the original software.
- **Integrity of the author’s source code** – The license may restrict source code from being distributed in a modified form if the license allows distribution of patches with the source code, so that patches can readily be distinguished from the base source code.
- **Distribution of license** – The rights of the license automatically pass through to all to whom the software is distributed (that is, the software can be redistributed to third parties without the permission of the original author).
- **Non-discrimination** – Usage cannot be denied to any person, group or field of endeavor.

These criteria have resulted in a plethora of licenses that can be used with terms that embrace a broader definition of open source software. A number of these licenses specifically reject the viral nature of the GPL while still requiring source code to be made available. These licenses arose from the desire to ensure the continued involvement of commercial organizations

in open source projects. Also, these licenses recognize that in some cases a commercial organization may not own all the rights to the code intended for public release – for example, where the existing code includes embedded software or other third party software subject to license terms that do not permit release under the terms of the GPL.

In addition, there are many other licenses that deal with open source but have not been approved by the Open Source Initiative because they do not comply with all the requirements of the Open Source Definition. One example is the Sun Community Source License, which is not available to everyone as it only pertains to countries that Sun believes offer reasonable prospects of intellectual property protection and enforcement.

The variety of open source licenses provides many opportunities for organizations to participate in the open source movement, whether to seek the benefits of the work of others, make contributions or just borrow code. A table comparing common open source licenses appears on pages 78-79. Two important items to highlight are:

- **BSD License** – Developed by the University of California at Berkeley, this license requires the developers to receive proper credit but permits modifications of open source code to be turned into proprietary, closed source code. Thus the BSD does not contain the copyleft restrictions found in GPL-style licenses.
- **Mozilla Public License** – Open source files (referred to as “covered” files) can be used with a commercial entity’s own files provided the covered files are not modified. If the covered files are modified, those modified files must be distributed as MPL files. The MPL enables proprietary and open source software to work together.

It is not uncommon to find articles comparing the virtues of the various open source licenses and acknowledging the complexities that arise. To wit: a commercial organization may modify open source software and license it back to the author but retain the rights to use that modified code in a proprietary application. In addition, from a legal perspective it

is not always clear whether one work is derived from, or is a modification of, another work.

In addition to license variations, another approach to encourage commercial involvement is the dual-license model typified by MySQL AB, Trolltech AS and Sleepycat Software. Under this model vendors offer their products under both an open source license and a commercial license. This allows open source projects to use the software at no cost (thus encouraging widespread use and testing), while organizations that wish to redistribute the software as part of a commercial product can do so by purchasing a commercial license. The organization is not required to publish or otherwise disclose its source code. Under the dual-license model, then, a product can be simultaneously licensed on different terms to different users for different purposes (e.g., under the GPL and under a commercial license).

From a license perspective, another way to illustrate the impact of the various open source software licenses is depicted in the inverted triangle on page 80. The triangle shows how both GPL-style licenses and proprietary licenses require strict compliance with their respective terms, notwithstanding that the two types of licenses are at opposite ends of the triangle on the issue of ownership and availability of their respective code.

### What Happened to Warranties?

Users of software products expect a number of (albeit limited) warranties from the licensor, for they give the user some comfort by way of recourse in certain circumstances.

However, it is not unusual for the license terms that govern a person’s use of open source software to be minimal to lacking in terms of warranties, specifying that:

- the software is provided on an “as is” basis
- all warranties relating to use or performance are disclaimed
- there is an exclusion of any liability (or liability is severely limited) by the licensor
- no indemnity is provided by the licensor against claims of intellectual property infringement by third parties brought against the customer.

```
void addChild(D
pper oldJspServ
() {child insto
throw new Ill
(sm.getStr
pper wrapper =
lean isJspServ
(isJspServlet)
oldJspServlet
if (oldJspServ
removeChild
```

# Major Open Source Licenses

	GNU General Public License (GPL) Version 2	GNU Lesser General Public License (LGPL) Version 2.1	BSD License July 1999 version
Use of License	License can be used for any software; however, it may not be modified.	License can be used for any software; however, it is designed primarily for use with code libraries. The license may not be modified.	License can be used for any software.
Fees	The licensee may charge a fee for the physical act of providing its customers with a copy, and may also charge a fee for any additional warranty protection offered.	The licensee may charge a fee for the physical act of providing its customers with a copy, and may also charge a fee for any additional warranty protection offered.	No express provisions.
Redistribution	Redistribution (including any modifications) must be on the terms of the GPL.	Redistribution (including any modifications) must be on the terms of the LGPL or the GPL.	Redistribution provided conditions in BSD license are met.
Availability of Source Code	Any redistribution must be accompanied by source code, or the licensee must offer to provide source code on request for a period of at least three years.	Any redistribution must be accompanied by source code, or the source code must be accessible from the same place as the object code.	No requirement to make source code available.
Application to Other Software	Any work that in whole or in part contains, or is derived from, the licensed software must be distributed under the GPL, <i>but</i> independent and separate works that are not derived from the program need not be so licensed. The mere bundling of separate software on the same distribution medium does not bring the separate software under the GPL.	A “work that uses the library” by being compiled or linked with it, and which contains no derivative of any portion of the library, falls outside the license. However, where software contains portions of the library as a result of such compilation or linking, that software may be distributed under any license terms provided that they permit modification for the customer’s own use and reverse engineering for such purposes.	No express provisions.
Warranties and Liability	No warranties and complete exclusion of liability to the extent permitted by law. The licensee can offer warranties to its own customers if it chooses to do so.	No warranties and complete exclusion of liability to the extent permitted by law. The licensee can offer a warranty to its own customers if it chooses to do so.	No warranties and complete exclusion of liability.
Governing Law	None	None	None
Termination	License automatically terminates on breach by the licensee of its terms.	License automatically terminates on breach by the licensee of its terms.	No express termination provisions.

Source: CSC with Gilbert & Tobin, Australian Attorneys

Mozilla Public License (MPL) Version 1.1	IBM Public License Version 1.0	Apache License Version 2.0
License may be modified and applied to other software provided it is renamed and otherwise makes clear that the new license contains different terms.	License can only be used for the licensed IBM software and contributions.	License can be used for any software. However, Apache Software Foundation projects must use this license and there is no dual license model.
The licensee may charge a fee for any additional warranty, support, indemnity or liability obligations offered.	No express provisions.	The licensee may offer a fee for warranty, support, indemnity or liability obligations to its customers.
Modifications must be licensed under the MPL. Covered code (original licensed software and any modifications) not released as source code can be distributed under different license terms, provided the license complies with the MPL.	If distributed in source code, must be under this license agreement. If distributed in object code, may be under a license of the licensee's choosing, although that license must then comply with certain conditions.	Redistribution must be on the terms of this license (including compliance with notice requirements, and the right to license modifications on different but consistent terms and conditions).
All modifications <i>must</i> be made available in source code, either on the same media or separately available. For the latter, modifications must remain available for at least 12 months from becoming available or six months after a subsequent version has been made available.	Any object code redistribution must be accompanied by the source code.	No requirement to make source code available.
The licensee may combine the software with other programs, in which case these other programs are not affected by the MPL. However, the licensee must still comply with the MPL for all covered code (original licensed software and any modifications).	The license expressly states that it does not apply to: (a) separate software modules distributed with the licensed software under their own license terms, or (b) any software that is not a derivative work of the licensed software.	No express provisions.
No warranty. Total exclusion of liability, except liability for death or personal injury resulting from a party's negligence to the extent applicable law prohibits such limitations.	No warranties and complete exclusion of liability. Where the licensee includes the software in a commercial product, the licensee must indemnify any other software contributors against certain third party claims.	No warranty and exclusion of liability for indirect or consequential damages to the extent permitted by law.
California	New York	None
License automatically terminates if license is breached and licensee fails to remedy within 30 days of becoming aware of the breach (all properly granted sublicenses survive). The license also provides for termination where the licensee brings a patent infringement claim against a contributor.	License automatically terminates if: (a) license is breached and the licensee fails to remedy within a reasonable period after becoming aware of the breach; or (b) the licensee institutes patent infringement proceedings against a contributor.	No express termination provisions. However, license automatically terminates if licensee institutes patent infringement proceedings against any entity.

```

self.supports
return* CHFaw
dered = self
* RESPONSE:
RESPONSE.s
return* rend
ler*(self, c
not* self.pr
f.setPreRen
* self.page
ender*(self
ear_cache:
nsaction(),
stPreRender
eclareProtect
rCache*(self
y clear out
stPreRender
isattr(self,
attr(self,*
attr(self,*
REQUEST: REQU
DtmlIfNeede
elf.dtmlAllo
eclareProtect
*(self, cool
se this pag
k.acquire()
f._v_blocks
f._v_cooked
/;
klock.relea
uatePreRend
* DTMLDocum
lerMidsection
string.find(
/; doc, disc
cept* Value
*return* f
string.find
; discussion

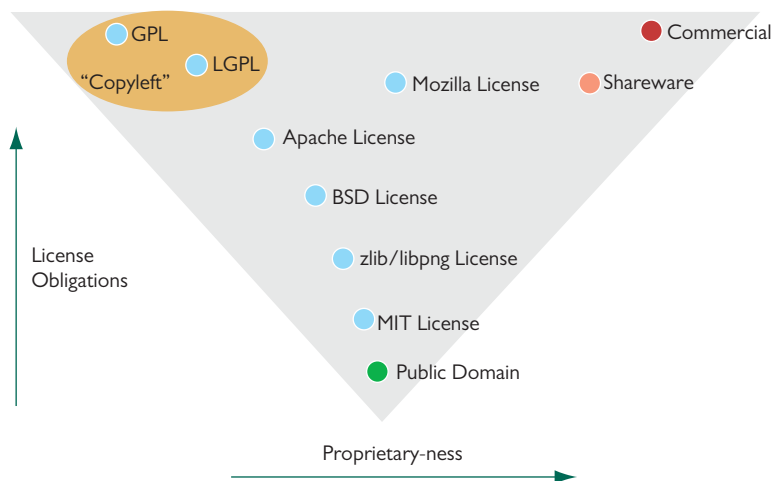
```



## SOFTWARE LICENSE OVERVIEW

In this overview of software licenses, the three corners of the triangle represent the primary license types: GPL, commercial and public domain. As you move away from the corners, you see license variations. GPL and commercial licenses carry the most license obligations but are at opposite ends of the proprietary scale regarding source code ownership and availability. Public domain software has no license at all.

Source: CSC



This circumstance arises because the very size of the user community can make it impossible to identify what attorneys refer to as the “chain of title” – the transparent history that determines ownership of a software product. Without a warranty from the licensor that it has the rights to license the product to a user and an indemnity to protect the user against a claim brought against the user by a third party, the user has no legal recourse should the user suffer damage as a result of being sued for (wrongfully) using the software.

In addition, the user is left to defend on its own any third party claim alleging infringement of the third party’s intellectual property rights.

Finally, the user has no remedy if use of the software fails to meet the user’s needs or expectations.

The absence of any warranties and indemnity for third party claims highlights a significant difference from the more traditional software licensing model. This applies equally to obscure and more mainstream open source software. (For example, see the IBM Public License Version 1.0 in the table on page 79.)

It should be noted that the numerous exclusions from liability or responsibility expressed in the various open source software licenses may mean that such agreements are not enforceable.

For example, an open source license is a contract and requires the giving of “consideration” to be enforceable. However, where one party to the contract undertakes to do something, but then excludes its liability for failing to do so, there may in fact be no promise at all (and therefore no real consideration). If this was a correct view, then the contract would never come into effect and the licensor would be deprived of the benefit of the exclusionary provisions.

Another issue that applies principally to GPL-style licenses arises because the license states that its terms apply automatically to a person who distributes code that contains GPL source code, whether or not the distributor was aware of the GPL terms or even that a small percentage of open source code is embedded in the product. This concept conflicts with many basic principles of contract law in various countries.

### The SCO Case

*SCO v IBM* should not necessarily be seen as a case that arises only because its subject matter relates to use of code to create open source software. (In this case, SCO alleges that a portion of SCO’s UNIX code has been copied by IBM and incorporated into Linux). This is essentially a case concerning a claim by SCO of breach of contract on the basis that its intellectual property rights have been infringed by IBM or by a breach of confidential information by IBM.

IBM counter-sued, ironically, asserting that SCO was in breach of the GPL and had, in turn, breached certain IBM patents. This resulted in an additional counter-claim by SCO that included a claim that the GPL was unconstitutional under U.S. law because its terms were inconsistent with U.S. copyright law.

SCO has since upped the ante by commencing legal action against end users Daimler-Chrysler and AutoZone, whom SCO alleges have been using the Unix code that is the subject of SCO's claim against IBM.

Should the SCO case ever get past the initial jousting stage and be heard in court, the validity of the GPL as a legitimate and enforceable set of terms and conditions may be determined.

However, the case highlights the ever-present risk of third party infringement claims based either on breach of copyright or infringement of a patent right. As open source software licenses like the GPL specifically disclaim any liability for infringement of intellectual property rights, it is entirely plausible that a developer could copy existing proprietary code (thereby infringing the copyright of the owner) and use it as the basis of an open source solution, which in turn is widely distributed in the marketplace.

In fact, a company in the United States, Open Source Risk Management LLC, has been established to offer insurance against what it deems a near-certainty: that lawsuits similar to SCO's will be brought against Linux vendors and users.

### The GPL in Court

Recently, in what may be the first judicial decision about the GPL, a Munich District Court granted a preliminary injunction in April 2004 against Sitecom Germany GmbH as sought by the netfilter/iptables open source project. Sitecom was offering a wireless access router product based on software licensed under the GPL and developed by the netfilter/iptables project.

According to the court order, Sitecom did not fulfill the obligations imposed by the GPL that applied to the netfilter/iptables software, as Sitecom refused to make its source code offering available to the user

community and refused to make its product available to users under the terms of the GPL.

The injunction prevents Sitecom from distributing its product unless it complies with the obligations imposed by the GPL. At the time of writing, it was not clear how Sitecom would respond.

### COMMERCIAL CONSIDERATIONS

Does the SCO case mean it is unsafe to use a solution that incorporates open source software? Absolutely not. What the SCO case reminds us is that there is often complexity around who actually owns the copyright in software code.

### Indemnity Plans for Open Source Software

With respect to the actual claims made by SCO, and to address the fears of Linux licensees, some distributors such as HP have announced that they will provide an indemnity to new (and existing ) customers who purchase Linux from them, provided the customer does not make any unauthorized changes to the Linux source code. In addition, the customers have to agree to only update their operating systems with changes made by their Linux provider, such as Red Hat or Novell/SUSE. HP felt the need to provide this indemnity to give its customers the comfort necessary to continue with Linux deployments.

Similarly, Sun has agreed to indemnify its customers who use its Linux-based Java Desktop System and its Solaris operating system.

It is likely that HP and Sun feel reasonably comfortable that SCO's case is unlikely to succeed. If that is not the case, the financial impact on HP and Sun could be considerable.

### Proprietary Licenses that Restrict Indemnity

There has been a recent trend for proprietary software vendors to vary their standard license terms to exclude liability for any embedded software that may be contained in their software product, whether or not the embedded software had its origins as open source software.

```
applicat
}
fireContain

public void addCh

Wrapper oia

if (!child
throw new
(sw.d

)

Wrapper wrap
boolean isJsp
if (isJspSer
```

For example, one leading OEM states in its software license agreements that it has no obligation to indemnify the licensee for a claim by a third party alleging infringement of its intellectual property rights, where the claim is based on “third party code, including but not limited to, open source code.”

Now, if the licensee knows that it is licensing open source software, such an exclusion may be reasonable. However, for a licensor to make the licensee assume complete risk for whether or not there is any embedded software in the licensor’s product seems to be nonsensical and should be rejected by the licensee. Otherwise, one of the fundamental protections that a licensee should receive from the party who is in the better position to know the content of its software will have been lost.

### Accountability

As noted in the Treasure Chest, customers want innovative solutions that are rich in functionality, minimize costs and are available quickly. This puts considerable pressure on the software company or service provider.

Some customers are unlikely to accept support that relies on developer mailing lists, newsgroups or peer user groups. These customers expect suppliers to be accountable and to stand by their deliverables. So although open source software is usually provided on an “as is” basis, a software company or service provider that delivers a solution incorporating open source software can expect its customers will want not only warranties covering the performance of the delivered solution but also a commitment to ongoing support and an indemnity against third party infringement claims.

With no “back-to-back” warranties or indemnities from the open source community to protect those who develop with open source software, the software company or service provider needs to accept that it is assuming a certain amount of potential exposure for which it cannot seek adequate redress.

### Government Perspective

Certainly there is global interest at the government level in adopting open standards and, at times, specifically endorsing the use of open source solutions. This is usually justified on the basis that governments wish to be vendor independent.

A recent example is the decision by the city of Munich to deploy Linux and other open source desktop applications across its 14,000 computers in its public administration.

The U.S. Department of Defense accepts open source software solutions for projects undertaken for the DoD. However, in a memo dated May 28, 2003, John Stenbit, Assistant Secretary of Defense and DoD’s Chief Information Officer, noted that open source software licensed under the GPL included restrictions that would still need to meet government policy. The implication is that GPL code has the potential to conflict with DoD requirements. Accordingly, Stenbit’s memo made it clear that a vendor must ensure that any open source software solution complies with all lawful licensing requirements:

As licensing provisions may be complex, the DoD Components are strongly encouraged to consult their legal counsel to ensure that the legal implications of the particular license are fully understood.<sup>38</sup>

### POLICY CONSIDERATIONS

It is clear that if open source code is used, the software development organization must fully document and retain the knowledge concerning that code.

Why is this all necessary? Because it would be naive to believe that every employee of a software development organization is meticulous in tracking the origin of all code that person uses in providing a deliverable. Why create code when existing code can be re-used? With easy accessibility to so much open source code, literally at the touch of a key on the keyboard, the likelihood of open source code finding its way into proprietary code is hardly far-fetched.

An internal approval process involving technical and quality assurance, along with sign-off from business and legal representatives, is necessary. This approval process needs to:

- Have the ability to capture and identify the proposed use of any open source software.
- Consider why there should be use of that particular open source code.
- Determine whether the use is consistent with business strategies of the organization.
- Ensure that any employees or contractors writing code for the organization have agreed in writing that the product of their labor will be owned by the organization.
- Confirm the code has previously been approved internally for use with respect to both technical specifications and license terms.
- Identify who will monitor ongoing compliance with applicable license terms, support and quality issues.

In addition, if the code has not been approved, then it may be prudent that:

- Legal advisors review the license terms.
- The bid or delivery manager reviews the business issues surrounding the use of the proposed code as part of a solution.
- Quality assurance, delivery assurance or the equivalent group determines the likely costs of support and, where appropriate, provides sign-off on the technical issues.

Perhaps historically not that much attention was paid to the intellectual property pedigree of code used within a software solution. But with ready access to open source code and the recent SCO litigation, organizations are becoming much more focused on the origin and conditions of use that might apply to code within their software solutions.

Not surprisingly, a potential auditing and management tool for detecting and managing proprietary and open source code has arisen to assist in maintaining quality control of the use of open source within an organization.

Although its product is still in development, Black Duck Software claims to have a tool that can “identify open source code residing in development projects and recognize potential licensing problems....[so] you can enjoy the advantages of open source software while mitigating management challenges and IP [intellectual property] risks....”<sup>39</sup>

Will it be able to decipher an entire source tree and determine that certain lines are from, say, an Eclipse project licensed under the Common Public License and other lines are a module governed by the GPL? Time will tell, but one can reasonably expect that the open source community itself will find a solution.

Though the existence of open source software is well known in the IT industry, the obligations that attach with its use are not. In particular, the GPL – though the source of much software innovation – may have a detrimental impact on commercial organizations whose business is the licensing of proprietary software. Such companies must understand the fine print and put in place appropriate processes to vet any intended use of open source software.

In addition, software companies and service providers must realize that ultimately they will have to take on the risk and provide warranties to any customer they provide a software solution to that incorporates open source software. Customers will not be interested in altruistic notions of free or open software; customers just care that the software works.

Open source solutions are safe to use despite the SCO case. For industry and government organizations seeking to reduce ongoing IT costs or looking for new applications with richer functionality, the opportunities with open source are virtually limitless. All potential open source customers must understand the open source software and licenses they intend to use, particularly when undertaking development in-house. They can also rely on their trusted service provider to ensure compliance – that is, leave it to the experts.



## GETTING STARTED



















































Open source is here to stay. Organizations around the world are using open source software. Businesses, governments and even software vendors are weighing in, backing the movement.





Starting with the software infrastructure, where open source software dominates, and moving up the stack, numerous open source software products are meeting growing acceptance, especially for operating systems, database management systems and consumer software. Commercial Linux distributions deliver out-of-the-box

desktop software for consumers and come bundled with office and multimedia software that was once available only in the Microsoft environment.

Open source offers an alternative, one that organizations need to consider in their IT strategy. Open source may not be for every situation, but in that case organizations should understand why. Organizations must examine the business value of open source (see Business Value of Open Source chart below) and look closely at their IT infrastructure and development processes.

## BUSINESS VALUE OF OPEN SOURCE

	Cost Reductions	Technology Transparency	Security/Risk Management	Time to Market	New Opportunities
Culture of Community					
Moving Up the Stack					
Mission Critical					
Sweet Spot					
Software Revolution					
At Your Service					
Invisible Man					
Market Force					
New Domains					
Fun Factor					

  
 least value                      most value

- \* cost reductions (outsource, open source, commodity computing, etc.)
- \* technology transparency (easier to integrate, interoperate and adapt)
- \* security/risk management (higher availability due to lower vulnerability)
- \* time to market (faster deployments = competitive edge)
- \* new opportunities (new business possibilities with open source)

The open source value proposition varies by organization and situation. One way to assess the business value of open source is to examine the report trends against five key business drivers. This matrix, while not an exact science, helps organizations focus on the areas of greatest opportunity by illustrating which trends play strongest in which areas.

Source: CSC

## FROM A TECHNICAL VIEW:

- Build expertise and relationships with open source vendors and the open source community.
- Encourage developers and system administrators to get involved with open source projects that relate to their jobs.
- Create a “safe list” of open source software for the organization to use or consider, and keep the list current.
- Launch trials of the most promising opportunities; start with non-critical systems.
- Assess open source software for interoperability.
- Assess open source software for platform independence.
- Design security in from the start.
- Develop policies for promoting and using open source software (e.g., policies for infrastructure, software development, mission-critical applications, security applications).
- Consider contributing software to open source projects to build trust and improve your credibility in the community. This will greatly improve the community’s willingness to provide support for you and your ability to tap the community’s resources.

Open source is an opportunity for service providers, who are well-positioned to play the role of trusted support provider and fill the accountability gap. CSC is committed to the open source movement; the company has many products, services, individuals and engagements that have made extensive use of open source software, including, in some cases, contributions back to the open source community.

- CSC does not view open source as an either-or battle with proprietary software vendors; rather, open source is about understanding the value that a higher degree of openness and collaboration can bring and applying that to the business.

For consumers, producers and service providers, the time is ripe for getting started with open source. It is a business decision, one to be factored into IT strategy and the software mix. Wise companies will manage and leverage open source, not ignore it. Open source is here to stay. Open source is open for business.

## NOTES

- 1 Christopher Koch, "Your Open Source Plan," *CIO Magazine*, March 15, 2003. <http://www.cio.com/archive/031503/opensource.html>
- 2 "Open Source in the Running for 30 Percent of New Systems," Gartner Symposium ITXpo 2003, March 10, 2003. <http://symposium.gartner.com/story.php?id.3390.s.5.html>
- 3 John D. Wolpert, "Breaking Out of the Innovation Box," *Harvard Business Review*, August 2002, p.4 (reprint).
- 4 For an in-depth discussion of the network effect see "The Architecture rEvolution: Exploring the Network Effect on Infrastructure, Applications and Business Process," CSC LEF Report, 2003. [http://www.csc.com/aboutus/lef/mds67\\_off/index.shtml#reports](http://www.csc.com/aboutus/lef/mds67_off/index.shtml#reports)
- 5 "The Cathedral and the Bazaar," Eric S. Raymond, September 11, 2000. <http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html>
- 6 "A Brief History of Hackerdom," Eric S. Raymond, in *Open Sources: Voices from the Open Source Revolution*, January 1999, p.6. <http://www.oreilly.com/catalog/opensources/book/raymond.html>
- 7 Ibid.
- 8 "The Boston Consulting Group Hacker Survey," Release 0.73, in cooperation with the Open Source Developer Network, July 24, 2002. <http://www.bcg.com/opensource/BCGHackerSurveyOSCON24July02v073.pdf>  
Note that 15 percent checked "other" for current occupation. Figures do not add to 100 percent due to rounding.
- 9 Flashmap Systems is a leader in creating intuitive, visual frameworks for modeling enterprise IT architectures. These frameworks depict a well-regarded and accepted taxonomy of IT product categories. <http://www.flashmapsystems.com/company.htm>
- 10 DCML "About" page. <http://www.dcml.org/aboutDCML.asp>
- 11 OfficeOffice.org is the official name of the application and the project.
- 12 Dan Tynan, "Hassle-Free E-Mail," *PC World*, March 2004 (see chart). <http://www.pcworld.com/reviews/article/0,aid,114149,pg,3,00.asp>
- 13 "The AP Relies on MySQL for Transaction-Heavy News Delivery System," June 24, 2003. [http://www.mysql.com/press/user\\_stories/ap.html](http://www.mysql.com/press/user_stories/ap.html)
- 14 Bruce Schneier, Crypto-Gram Newsletter, September 15, 1999. <http://www.schneier.com/crypto-gram-9909.html#OpenSourceandSecurity>
- 15 Whitfield Diffie, "Risky business: Keeping security a secret," *ZDNet*, January 16, 2003. <http://zdnet.com.com/2100-1107-980938.html>
- 16 "Open Source Databases: All Dressed Up, Only So Many Places to Go," Aberdeen Group White Paper, March 2004, pp. 7-10. [www.aberdeen.com/ab\\_abstracts/2004/03/03040006.htm](http://www.aberdeen.com/ab_abstracts/2004/03/03040006.htm)
- 17 James Maguire, "IBM Developing Open-Source Server Software," *NewsFactor Network*, December 23, 2002. [http://www.newsfactor.com/story.xhtml?story\\_id=20322](http://www.newsfactor.com/story.xhtml?story_id=20322)
- 18 "Questions and Answers on Open-Source Licensing," Gartner Research Note, October 28, 2002, p.1.
- 19 "The Open Source Paradigm Shift," Tim O'Reilly, presentation, June 2003, slide 20. <http://tim.oreilly.com/opensource/ParadigmShift.pdf>
- 20 W. Brian Arthur, "Is the Information Revolution Over?" *Business 2.0*, March 2002.
- 21 "Embedded Systems Development Trends: Asia," *EE Times - Asia* – Gartner/DataQuest White Paper, October 2003, pp. 20-21.
- 22 "From Server Room to Living Room: How open source and TiVo became a perfect match," *QUEUE*, July/August 2003, p.25.
- 23 "Device Profile: Dreambox DM7000 – An Open TV Hacker's Paradise," LinuxDevices.com, October 2003. <http://linuxdevices.com/articles/AT7482684956.html>
- 24 "Your Open Source Strategy," Forrester Research, September 2003, p.1.
- 25 "Free and Open Source Software," Statskontoret, the Swedish Agency for Public Management, 2003, p. 4.
- 26 H.G. Wells, "World Brain: The Idea of a Permanent World Encyclopaedia," contribution to the new *Encyclopédie Française*, August 1937. [http://sherlock.berkeley.edu/wells/world\\_brain.html](http://sherlock.berkeley.edu/wells/world_brain.html)
- 27 MIT Open Source Building Alliance prospectus, November 11, 2003, p.1. [http://architecture.mit.edu/house\\_n/web/projects/OSBAProspectusNov11-2003.pdf](http://architecture.mit.edu/house_n/web/projects/OSBAProspectusNov11-2003.pdf)
- 28 Anna Salleh, "Push to free up biotech tools for all," *ABC Science Online*, December 1, 2003. [www.abc.net.au/science/news/stories/s999733.htm](http://www.abc.net.au/science/news/stories/s999733.htm)
- 29 "Australian genetics pioneer recognized in global top 50," Cambia press release, November 25, 2003. [http://www.cambia.org/downloads/global\\_top\\_50.pdf](http://www.cambia.org/downloads/global_top_50.pdf)
- 30 Graeme O'Neill, "Open-source biology stance earns international honour," *Australian Biotechnology News*, March 12, 2003. [http://www.cambia.org/downloads/Biotechnology\\_News\\_Dec\\_03.pdf](http://www.cambia.org/downloads/Biotechnology_News_Dec_03.pdf)
- 31 Public Library of Science home page. <http://www.plos.org/index.html>
- 32 "Public Library of Science Wins Wired Magazine Science Rave Award," PLoS press release, March 18, 2004. [www.plos.org/news/announce\\_rave.html](http://www.plos.org/news/announce_rave.html)
- 33 "The Architecture rEvolution: Exploring the Network Effect on Infrastructure, Applications and Business Process," CSC LEF Report, 2003. [http://www.csc.com/aboutus/lef/mds67\\_off/index.shtml#reports](http://www.csc.com/aboutus/lef/mds67_off/index.shtml#reports)
- 34 PlaneShift "About" page. <http://www.planeshift.it/about.html>
- 35 PlaneShift License page. <http://www.planeshift.it/pslicense.html>
- 36 J.D. Hildebrand, "A Road Less Traveled," *Software Development Times*, January 1, 2001. [http://www.sdtimes.com/cols/opensourcewatch\\_021.htm](http://www.sdtimes.com/cols/opensourcewatch_021.htm)
- 37 GNU Lesser General Public License, Preamble. <http://www.opensource.org/licenses/lgpl-license.html>
- 38 Memorandum by John P. Stenbit, CIO, U.S. Department of Defense, May 28, 2003. [http://www.egovos.org/rawmedia\\_repository/822a91d2\\_fc51\\_4e6e\\_8120\\_1c2d4d88fa06?document.pdf](http://www.egovos.org/rawmedia_repository/822a91d2_fc51_4e6e_8120_1c2d4d88fa06?document.pdf)
- 39 Black Duck Software home page. <http://www.blackducksoftware.com/>

## APPENDIX: HANDY WEB SITES

Many Web sites have extensive information on the topics discussed in this report. Here are a few sites to get you started.

### CULTURE OF COMMUNITY

GNU

[www.gnu.org](http://www.gnu.org)

“The Cathedral and the Bazaar”  
(paper by Eric Raymond)

[catb.org/~esr/writings/cathedral-bazaar/  
cathedral-bazaar/index.html](http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html)

SourceForge

[www.sourceforge.net](http://www.sourceforge.net)

GForge

[www.gforge.org](http://www.gforge.org)

### MOVING UP THE STACK

Flashmap Systems

[www.flashmapsystems.com](http://www.flashmapsystems.com)

BSD

[www.bsd.org](http://www.bsd.org)

Globus

[www.globus.org](http://www.globus.org)

Sun Microsystems’ Grid Engine

[gridengine.sunsource.net](http://gridengine.sunsource.net)

OpenSSL

[www.openssl.org](http://www.openssl.org)

OpenLDAP

[www.openldap.org](http://www.openldap.org)

Apache Software Foundation

[www.apache.org](http://www.apache.org)

Open Source Applications Foundation

[www.osafoundation.org](http://www.osafoundation.org)

JBoss

[www.jboss.org](http://www.jboss.org)

MySQL

[www.mysql.com](http://www.mysql.com)

OpenCms

[www.opencms.org](http://www.opencms.org)

TikiWiki

[tikiwiki.org](http://tikiwiki.org)

Covalent Technologies

[www.covalent.com](http://www.covalent.com)

GNOME Foundation

[www.gnome.org](http://www.gnome.org)

OpenOffice

[www.openoffice.org](http://www.openoffice.org)

Compiere

[www.compiere.org](http://www.compiere.org)



```
*if* *not*  
self.s  
*return* s  
  
*def* *preRead  
*if* *clear  
get_trans  
self.setPr  
  
security.decl  
*def* *transPa
```

continued from page 87

## MISSION CRITICAL

Google

[www.google.com](http://www.google.com)

Associated Press

[www.ap.org](http://www.ap.org)

Mars Science Activity Planner

[robotics.jpl.nasa.gov/people/jnorris/SAP-IEEEAS03.pdf](http://robotics.jpl.nasa.gov/people/jnorris/SAP-IEEEAS03.pdf)

Credit Suisse First Boston

[www.csfb.com](http://www.csfb.com)

Danish Ministry of Finance

[www.fm.dk](http://www.fm.dk)

Deutsche Börse Group

[deutsche-boerse.com](http://deutsche-boerse.com)

Open Source Development Labs

[www.osdl.org](http://www.osdl.org)

## SWEET SPOT

Zeus Technology

[www.zeus.com/products/zws](http://www.zeus.com/products/zws)

“Open Source Databases: All Dressed Up, Only So Many Places to Go” (Aberdeen Group White Paper)

[www.aberdeen.com/ab\\_abstracts/2004/03/03040006.htm](http://www.aberdeen.com/ab_abstracts/2004/03/03040006.htm)

Metapa

[www.metapa.com](http://www.metapa.com)

Northwest Educational Technology Consortium's Open Options

[www.netc.org/openoptions/index.html](http://www.netc.org/openoptions/index.html)

## SOFTWARE REVOLUTION

Expect

[expect.nist.gov](http://expect.nist.gov)

H.E.A.T.

[www.heatscanner.com](http://www.heatscanner.com)

“John the Ripper”

[www.openwall.com/john](http://www.openwall.com/john)

PasTmon

[pastmon.sourceforge.net](http://pastmon.sourceforge.net)

OpenCyc

[www.opencyc.org](http://www.opencyc.org)

Intalio

[www.intalio.com](http://www.intalio.com)

ExoLab

[www.exolab.org](http://www.exolab.org)

## AT YOUR SERVICE

Red Hat

[www.redhat.com](http://www.redhat.com)

Novell

[www.novell.com](http://www.novell.com)

HP Linux Indemnity Site

[h10018.www1.hp.com/wwsolutions/linux/linuxprotection.html](http://h10018.www1.hp.com/wwsolutions/linux/linuxprotection.html)

Zope Community

[www.zope.org](http://www.zope.org)

Workspot

[www.workspot.com](http://www.workspot.com)

## INVISIBLE MAN

Wind River Systems  
[www.windriver.com](http://www.windriver.com)

Zultys Technologies  
[www.zultys.com](http://www.zultys.com)

TiVo  
[www.tivo.com](http://www.tivo.com)

Dream-Multimedia-TV  
[www.dream-multimedia-tv.de](http://www.dream-multimedia-tv.de)

PhatNoise  
[www.phatnoise.com](http://www.phatnoise.com)

D-Link Systems' Central Home Drive  
[www.dlink.com/products/?pid=283](http://www.dlink.com/products/?pid=283)

Stargate Project (includes Intel Research  
Personal Server)  
[platformx.sourceforge.net](http://platformx.sourceforge.net)

Land Warrior  
[www.fas.org/man/dod-101/sys/land/land-warrior.htm](http://www.fas.org/man/dod-101/sys/land/land-warrior.htm)

## MARKET FORCE

Microsoft Windows Template Library Project  
[sourceforge.net/projects/wtl](http://sourceforge.net/projects/wtl)

Openadapter  
[www.openadapter.org](http://www.openadapter.org)

Eclipse  
[www.eclipse.org](http://www.eclipse.org)

Hibernate  
[www.hibernate.org](http://www.hibernate.org)

## NEW DOMAINS

Wikipedia  
[www.wikipedia.org](http://www.wikipedia.org)

Openlaw  
[cyber.law.harvard.edu/openlaw](http://cyber.law.harvard.edu/openlaw)

ThinkCycle  
[www.thinkcycle.org](http://www.thinkcycle.org)

OpenCores  
[www.opencores.org](http://www.opencores.org)

IBM's Power Architecture  
[www-1.ibm.com/technology/power](http://www-1.ibm.com/technology/power)

Intel's Open Source Machine Learning Library  
[www.intel.com/ca/pressroom/2003/1208.htm](http://www.intel.com/ca/pressroom/2003/1208.htm)

Open Source Building Alliance  
[architecture.mit.edu/house\\_n/web/projects/OSBAProspectusNov11-2003.pdf](http://architecture.mit.edu/house_n/web/projects/OSBAProspectusNov11-2003.pdf)

Cambia  
[www.cambia.org](http://www.cambia.org)

Public Library of Science  
[www.plos.org](http://www.plos.org)

Open Spectrum  
[www.reed.com/dprframeweb/dprframe.asp](http://www.reed.com/dprframeweb/dprframe.asp)

Creative Commons  
[creativecommons.org](http://creativecommons.org)

MIT's OpenCourseWare  
[ocw.mit.edu/index.html](http://ocw.mit.edu/index.html)

The Open Source Cookbook: Fuel for Geeks  
[www.ibiblio.org/oscookbook](http://www.ibiblio.org/oscookbook)

continued from page 89

## FUN FACTOR

Ogg Vorbis CODEC Project

[www.xiph.org/ogg/vorbis](http://www.xiph.org/ogg/vorbis)

VideoLAN

[www.videolan.org](http://www.videolan.org)

Project Mayo

[www.projectmayo.com](http://www.projectmayo.com)

PlaneShift

[www.planeshift.it](http://www.planeshift.it)

NetHack

[www.nethack.org/index.html](http://www.nethack.org/index.html)

id Software

[www.idsoftware.com/](http://www.idsoftware.com/)

Neverwinter Nights

[nwn.bioware.com/about/index.html](http://nwn.bioware.com/about/index.html)

## LEGAL AND BUSINESS ISSUES

Free Software Foundation

[www.gnu.org/philosophy/  
philosophy.html#AboutFreeSoftware](http://www.gnu.org/philosophy/philosophy.html#AboutFreeSoftware)

GPL

[www.fsf.org/licenses/gpl.html](http://www.fsf.org/licenses/gpl.html)

More about the GPL

[www.gnu.org/licenses/gpl-faq.html](http://www.gnu.org/licenses/gpl-faq.html)

LGPL

[www.opensource.org/licenses/lgpl-license.php](http://www.opensource.org/licenses/lgpl-license.php)

Open Source Definition from OSI

[opensource.org/docs/definition.php](http://opensource.org/docs/definition.php)

Sun Community Source License

[www.sun.com/software/communitysource/j2se/  
java2/index.html](http://www.sun.com/software/communitysource/j2se/java2/index.html)

“An Overview of ‘Open Source’ Software Licenses”  
(report by the Software Licensing Committee  
of the American Bar Association’s Intellectual  
Property Section)

[www.abanet.org/intelprop/opensource.html](http://www.abanet.org/intelprop/opensource.html)

Open Source Risk Management

[www.osriskmanagement.com](http://www.osriskmanagement.com)

Black Duck Software

[www.blackducksoftware.com](http://www.blackducksoftware.com)

## ACKNOWLEDGMENTS



LEF Associates **Stefan Höhn** (far left) and **Gabor Herr** conducted the research for this report. Stefan is a lead architect and team leader of the J2EE and Open Source team, and Gabor is a solution architect on the team. Both work in CSC's e-Business and Technology Center in Wiesbaden, Germany. The two poured their insights and their heart and soul into the report, which reflects their passion and business sensibility around the open source movement. As the research came to a close, Stefan and Gabor were deep into their next client engagement, also involving open source. [shoehn2@csc.com](mailto:shoehn2@csc.com), [gherr@csc.com](mailto:gherr@csc.com)

Assisting Stefan and Gabor with significant contributions were Maja Kreikemeier, research; Tom Knapp, legal and business issues; and Lisa Braun, writing and editing.

The LEF would like to thank the many others who contributed to the research and review of this report:

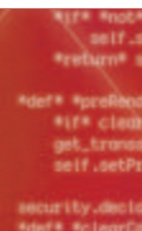
Tony Ardouin, CSC  
Jason Arnold, CSC  
Rolf Barth, Deutsche Börse Group  
Joerg Benischke, Deutsche Bank  
Dan Benyamin, PhatNoise  
Graham Bevan, CSC  
Derek Binney, CSC  
Anke Bramstedt, CSC  
John De Oliveira, Cypcorp  
Andrew Doble, CSC  
Tim Dooley, CSC  
Ken Ferguson, Huntsman Corporation  
Mark Foley, CSC

Aaron Fuller, CSC  
Ismael Ghalimi, Intalio  
Bruce Gritton, U.S. Navy  
Frank Habraken, CSC  
Graeme Hay, CSC  
Peter Henningsen,  
Danish Ministry of Finance  
Hans Jayatissa, CSC  
Charles Kramer, Esq.  
Jim Mundy, CSC  
Jim Petrassi, CSC  
Dave Powell, Metapa  
Joe Reed, CSC

Howard Smith, CSC  
Jason Snyder, CSC  
Bob Solis, CSC  
Juan Carlos Soto, Sun Microsystems  
Marc Stern, CSC  
Bob Tarling, CSC  
Jeff Tash, Flashmap Systems  
Michael Tiemann, Red Hat  
Paul Tocci, CSC  
Michael Uhlig,  
Delta Lloyd Deutschland AG  
Christian Wernberg-Tougaard, CSC  
Rolf Wilms, CSC

The report team used a variety of open source software in working on this report. The team leveraged the CSC Toolbox collaborative environment, which provided a shared drive (using WebDAV based on Apache) and a wiki. The wiki was an ideal platform for sharing research materials globally, as the contributors were from all over the world. Early in the project, the team created "mind maps" based on FreeMind. As the project progressed, the team used OpenOffice for giving presentations. Throughout the project, Mozilla Firefox was the team's browser of choice.





LEGO, MINDSTORMS and the LEGO logo are trademarks of The LEGO Group.

Motorola A760 and Motorola E680 are trademarks of Motorola, Inc.

Logos on cover (from right to left):

- |                        |  |
|------------------------|--|
| [Linux Penguin]        | The Linux Penguin “Tux” was created by Larry Ewing (lewing@isc.tamu.edu).  |
| [GNU Head]             | Copyright (C) 1996, 1997, 1998 Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA. Verbatim copying and distribution of this entire article is permitted in any medium, provided this notice is preserved. |
| [Apache Feather]       | Apache is a trademark of The Apache Software Foundation and is used with permission.   |
| [OpenOffice.org Birds] | OpenOffice.org birds are used with permission.   |
| [GNOME Foot]           | The GNOME Foot Logo on the cover is the property of the GNOME Foundation, and is used with permission.   |





## Computer Sciences Corporation

### Worldwide CSC Headquarters

#### The Americas

2100 East Grand Avenue  
El Segundo, California 90245  
United States  
+1.310.615.0311

#### European Group

The Royal Pavilion  
Wellesley Road  
Aldershot  
Hampshire GU11 1PZ  
United Kingdom  
+44(0)1252.534000

#### Australia/New Zealand

26 Talavera Road  
Macquarie Park NSW 2113  
Australia  
+61(0)2.9034.3000

#### Asia

139 Cecil Street  
#08-00 Cecil House  
Singapore 069539  
Republic of Singapore  
+65.6221.9095

### About CSC

*Computer Sciences Corporation helps clients achieve strategic goals and profit from the use of information technology.*

*With the broadest range of capabilities, CSC offers clients the solutions they need to manage complexity, focus on core businesses, collaborate with partners and clients, and improve operations.*

*CSC makes a special point of understanding its clients and provides experts with real-world experience to work with them. CSC is vendor-independent, delivering solutions that best meet each client's unique requirements.*

*For more than 40 years, clients in industries and governments worldwide have trusted CSC with their business process and information systems outsourcing, systems integration and consulting needs.*

*The company trades on the New York Stock Exchange under the symbol "CSC."*