



Toll Free: 1-888-670-8889 International 1-519-432-3550 www.infotech.com

Free and Open Source Software: A Field Guide

Free and open source software (FOSS) is rapidly gaining credibility with corporate decision makers. Use this field guide to find out more about the variety of FOSS options available and how you can put them to work in your day-to-day IT environment.

Table of Contents

Background	3
Introduction	3
History.....	3
The GNU Project and Free Software.....	3
Linus Torvalds and Linux	5
Open Source and the Corporate World.....	6
Field Guide Criteria.....	7
FOSS Operating Systems	8
GNU/Linux	8
GNU/Linux Distributions	8
GNU/Linux File/Print Servers.....	10
GNU/Linux on the Desktop.....	11
Other FOSS OSes – The BSDs	13
FOSS and the Internet	15
Basic Infrastructure.....	15
Web Scripting.....	15
Application Servers	16
Content Management Systems (CMS)	17
Software Development	18
Basic Tools.....	18
Advanced Tools	19
FOSS Database Management Systems.....	20
Middleware	22
Conclusion	22

Background

Introduction

Everyone working in the IT industry today has heard of Open Source Software and knows something about the impact it is having on how we develop and deploy software solutions. Yet there remains many misconceptions and misunderstandings about what exactly Free and Open Source Software (FOSS) is. Some people are unclear about what the difference is between Free software and Open Source software. Others think that FOSS has only limited applicability and is not relevant to their particular IT sector. Finally, even those familiar with FOSS may not be aware of the great variety and excellent FOSS options available. The purpose of this white paper is to educate the reader about FOSS and to clear up these and related misconceptions.

Despite the fact that FOSS has been embraced by such industry heavyweights as IBM, Hewlett-Packard, and Sun Microsystems, most IT managers still look at FOSS as something wild and a bit untamed. Hence the name of this white paper – a field guide. Field guides are useful little tools that help you spot and understand the variegated beasts out there in the wild. This field guide will give you some insights into how you might actually capture some of these FOSS creatures and put them to use in your day-to-day work environment.

History

The GNU Project and Free Software

The 20th anniversary of the Free Software movement was recently celebrated. It was on September 27, 1983 that Richard Stallman sent out the initial announcement (<http://www.gnu.org/gnu/initial-announcement.html>) of the GNU project. In it Stallman called for the creation of a totally free Unix-compatible operating system that he named GNU (for GNU is Not Unix). Self-referential names are the hallmark of geeks, and Stallman is the quintessential geek/hacker.

Free software, as Stallman defines it, has four characteristics:

1. The freedom to run the program, for any purpose.
2. The freedom to modify the program to suit your needs.
3. The freedom to redistribute copies, either gratis or for a fee.
4. The freedom to distribute modified versions of the program, so that the community can benefit from your improvements.

The mechanism Stallman created to ensure these freedoms is what he calls “copyleft” – the GNU General Public License (GPL). This license is structured in a way to guarantee the four freedoms Stallman advocates. You can read more details about the GPL and related licenses at the GNU Project Web server site at <http://www.gnu.org/gnu/the-gnu-project.html>.

When Stallman made his announcement, it was considered a totally quixotic task. He was seen by many as somewhat ridiculous. But precisely because Stallman was motivated by moral considerations, not financial ones, he was undeterred by the financial and technical difficulties he faced over the years. He continued on his merry way, and with the help of others, he created a large infrastructure of compilers, editors, and utilities that went a long way toward realizing his vision of a Unix replacement. You can read more about the history of the GNU Project at <http://www.gnu.org/gnu/the-gnu-project.html>.

Why Free Software Matters

Stallman’s motivation for undertaking the GNU project might seem strange to many people. He was not hoping to create a software product that would become the basis of a VC-funded company. Nor was he looking for fame or personal glory. He was taking a moral stand.

Stallman believes that software technology is the basis of information exchange in modern society. Since the free exchange of information is a fundamental right and the basis of democracy, it is essential that software be free and unencumbered as well. For Stallman, the important thing is that software be “free as in speech.” Perhaps because he is so uncompromising, Stallman makes many people uncomfortable. For example, many people are annoyed about his insistence on using the term GNU/Linux, which you can read about at <http://www.gnu.org/gnu/gnu-linux-faq.html>.

But Stallman’s contentions about freedom and software (<http://www.gnu.org/philosophy/why-free.html>) are rapidly becoming mainstream issues.

A few years ago, Congress passed a new copyright law known as the Digital Millennium Digital Act (DMCA). Whatever the economic merits of this law, copyright laws are now being used to limit criticism of corporate entities. In a very recent case, internal documents of the Diebold Corporation have been made public. These documents reveal serious and disturbing flaws in Diebold’s software for electronic voting systems. These documents are being published all over the Internet. Diebold is using the DMCA as a mechanism to force individuals and organizations to take these documents down from their Web sites.

Stallman argues, that if free software did not exist, corporations could build mechanisms into software that would allow them to restrict the dissemination of any information that is not to their liking. It is free software that makes it

impossible for Diebold and other corporations from restricting free speech. Stallman develops these ideas in a futuristic story (<http://www.gnu.org/philosophy/right-to-read.html>) about what our society might be like if there was no free software.

Linus Torvalds and Linux

We now fast-forward eight years. It's 1991. The IBM PC and Microsoft MS-DOS have spread throughout the corporate world. Microsoft has recently released Windows 3.0 and it is starting to have an impact. Windows NT is just about to be released in its first version.

Linus Torvalds, a soft-spoken hacker and second year student of Computer Science at the University of Helsinki, was looking for something more sophisticated and useful than DOS for his 386/486 PC. Specifically, he wanted a Unix clone. He started working on it in April of that year.

The Internet was already quite large at that time, although quite minuscule by today's standards. It was mostly used by people at universities who were the only one's to have the equivalent of what we call a broadband connection. The World Wide Web had not yet been created, but the Internet was still used to exchange information through e-mail and what was called news groups. On August 21, 1991 Torvalds posted an announcement in a news group about his project, asking for help. In September 1991, version 0.1 of what was to become Linux was released to the world.

Torvalds from the beginning made the critical decision to build Linux on top of Stallman's GNU infrastructure. His focus was on creating the operating system kernel - the heart of an operating system. Once he had a kernel, wedded with the GNU tools, a full-fledged Unix-like OS would finally exist that was totally free, both monetarily and in Stallman's sense of the word. Equally important, Torvalds adopted the copyleft GNU GPL for the Linux software license. This was to have enormous implications.

Torvalds understood that the GNU GPL was not just a software license, but also the basis of a software development method. When Stallman started the GNU project, his hope was to recreate the atmosphere of sharing and mutual co-operation that had existed in the software world prior to the rise of proprietary software. The GPL encourages sharing because it guarantees that no one party can benefit from the sharing at the expense of others. Whatever anyone contributes is shared equally by everyone. People happily contribute to Torvalds' project because they know no one could exploit their work at their expense. They also knew they equally benefit from the work of others.

The explosive growth of the Internet in the decade subsequent to Torvalds' initial posting helped accelerate the positive self-reinforcing growth of contributors to Linux, GNU, and many other free software projects.

Open Source and the Corporate World

A few years ago, a group of advocates, who had a more pragmatic perspective than Stallman, started the Open Source Initiative (<http://www.opensource.org/>). Open Source advocates stress the pragmatic “freedoms” this type of software provides to developers and organizations. First and foremost is the economic benefit of FOSS, which is often described by the humorous phrase “free as in beer.” While there are many costs associated with using software, including training, support, and deployment, licensing fees are one of the main costs. The fact that nearly all FOSS software does not have associated licensing costs is a huge economic benefit.

The FOSS Economic Model

In the early years of its adoption, many in corporate IT saw the FOSS model as (at worst) vaguely “communistic,” and at best tree-hugging hippy-like behavior. But Torvalds the pragmatist, and many after him, clearly understood the economic model of FOSS development. If developer X devotes 5 days a month to developing FOSS product A, perhaps s/he (or the employer) lost \$5,000 of opportunity cost in consulting work or salary. But s/he knows that 100 other developers are doing the same. So this \$5,000 investment has just yielded a return in software now worth \$500,000. The ROI of the FOSS model should make any VC envious.

As a side effect, even people who don't develop the software, that is who don't contribute directly, can get to use the software as well, most often without any payment. Proponents of proprietary software find this “free-rider” aspect most disturbing. But a rational economic model requires that one only calculate the benefit to oneself. The \$5,000 investment yields the \$500,000 software needed today, which without that investment would not have existed. It is irrelevant to the investor that tomorrow others who didn't invest the \$5,000 also benefit. In fact, investors do benefit from the “free-riders”, since these find bugs and test the software in varied environments, helping it become more secure and robust over time. So “free-riders” contribute value, although of a lesser sort. Finally, many investors start as “free-riders,” so they are an important part of the growth of the model.

Besides the economic benefits, Open Source advocates stress that FOSS is “free as in exchange of ideas.” They argue that software developed using this co-operative model will be inherently more secure and robust than proprietary counterparts. With many eyes looking at the code, with the “free-riders” checking the software on multiple hardware configurations, bugs and security holes are shaken out more quickly. Moreover, those who develop the software aren't driven by trying to sell anything. Rather they are trying to solve an actual problem they have. The availability of the

source codes, allows developers to build upon the ideas, of others without reinventing the wheel. Hence the software created is more tuned to users actual needs, not what Dr. Seuss called a *thneed*, or what proprietary software vendors call “features.”

Finally, Open Source advocates stress that FOSS software is “free as in markets” – i.e. it provides freedom from vendor lock-in. Vendor lock-in is relevant to all proprietary software, not just Microsoft products. Recently, I received a call from a consultant working for a local New York City hospital that had a patient database system. The database vendor’s license required that any software add-ons that access the database had to be developed by the vendor. The hospital desperately needed to build a bridge between the patient database and some other software component. They were hoping to use “screen scraping” software to get at the patient information without having to pay the vendor. Unfortunately I had to advise them that even that technique would be a violation of their contract. FOSS not only allows you to own your own data, it openly provides you with the code for the data formats and the methods that access that data. In all cases, your own IT team or any outside vendor can modify and maintain your software. Vendors compete on service and value, not by holding you hostage to their code.

Suffice it to say that Open Source advocates are less picky than Stallman about how they interact with non-Free Software. The term FOSS covers the whole gamut of software that is based on source code availability and unrestricted sharing (the politically correct stress the European origins of FOSS by using the term *Libre* and thus FLOSS, but we will avoid that term for obvious reasons).

Whatever the motivation for using FOSS, no one can deny this: 20 years after Stallman started the GNU project and 12 years after Linus Torvalds started work on Linux, the Linux kernel is about to hit version 2.6. GNU/Linux – the full operating system that uses Torvalds’ kernel and the GNU infrastructure, as well as FOSS contributed by many individuals and organizations – presents the greatest competitive threat to Microsoft. GNU/Linux has a real chance of soon dominating the IT server sector. Huge corporations like IBM are pouring in billions of dollars into Linux and other FOSS development. The economic and other benefits of FOSS are being universally recognized in the corporate world. With that in mind, it’s time to go out in the field, and look at what I once heard an IBM VP call the “FOSS ecosystem.”

Field Guide Criteria

This field guide is targeted at developers and IT managers who are looking for practical tools and solutions. Even when recognizing the benefits of FOSS, the biggest barrier to FOSS adoption is the fear that the product has inadequate support or that the product might stop being developed because of lack of resources. While not necessarily a barrier, IT managers are also concerned about the hidden costs of FOSS deployment. They note that FOSS products are often less packaged than their proprietary equivalents, and so require more up-front resources to deploy.

Hence, to ensure that the FOSS tools mentioned in this field guide are all low-risk choices and no more difficult to deploy than their proprietary equivalents, they must meet the following criteria:

- They are commercial grade products being used in Fortune 500 companies.
- They have excellent online documentation and support available.
- They have an active, funded, and growing development and user community.
- They have commercial support and/or training available.

To meet these criteria, the FOSS products found here are, for the most part, development, infrastructure, and middleware products. Desktop FOSS products are often less mature and most do not meet these criteria. However, IBM recently announced a major commitment to FOSS on the desktop, as has Sun Microsystems and Novell. Perhaps next year's field guide will include many more FOSS desktop products.

FOSS Operating Systems

GNU/Linux

We start with GNU/Linux simply because it is the most widely known and deployed FOSS product.

GNU/Linux Distributions

The most well known of the Linux distributions is the commercial distribution, Red Hat (<http://redhat.com/>). You can't go wrong using Red Hat. The company has a good understanding of the FOSS development model, gives back to the community, and does a good job of packaging GNU/Linux in useful forms. If your company needs good Linux technical support, that is mainly what Red Hat provides. Mandrake (<http://www.mandrakesoft.com/>) is another distribution with a sterling reputation, especially for being easy to install. It recently got top-ratings in an InfoWorld survey (http://www.infoworld.com/article/03/08/01/30TCLinux_1.html?s=feature) of enterprise GNU/Linux distributions.

SUSE (<http://www.suse.com/us/>) is the European equivalent of Red Hat, and hence is more popular in Europe than in North America. However, Novell (<http://www.novell.com/>) recently announced the purchase of SUSE (along with an investment of IBM in Novell), so it is sure to grow its presence.

This author's personal favorite, however, is Debian (<http://www.debian.org/>). Unlike the others, Debian is not a company and does not directly provide commercial support (although commercial support for Debian is available). However, Debian takes a sophisticated project

management approach to tailoring GNU/Linux distributions towards different needs. If used properly, it can lessen the need for technical support.

GNU/Linux Technical Support

Licensing fees are a controversial topic in the FOSS world. Often the exact same product is available for free, so why pay the licensing fee? In very large corporations, there is some internal logic that compels them to do this. The finance department needs someone to “blame” if there is a problem, so signing a licensing contract makes sense. But for small- and medium-sized businesses (SMBs), the logic is less compelling. What FOSS vendors *do* provide for all customers is training and support. But you don’t need a *licensing* contract to get that.

If your company is very large and/or has very sophisticated needs, Red Hat or SUSE probably are good alternatives for support and/or training. For SMBs’ day-to-day system administration problems, the large commercial GNU/Linux vendors may be overkill. Since GNU/Linux can easily be managed remotely and does not require large amounts of time, especially if you use the Debian platform), outsourcing/time-sharing system administration is a viable option even for the smallest company. One GNU/Linux sysadmin can manage far more systems with significantly less overhead than their Window’s equivalent. Small and local companies or independent contractors can provide affordable support options.

If you have in-house IT staff unfamiliar with the GNU/Linux platform, migration to GNU/Linux probably will require outside help and staff training. But once the migration has been completed, the IT staff has been trained, and the migration successfully completed, there are many excellent free or inexpensive resources available to deal with simple issues. Many support issues can be solved by resources on FOSS product community sites or by referring to one of the many good administration books available for GNU/Linux. You can even get online access to full text versions of many of these books at the Safari Bookshelf (<http://safari.oreilly.com/>). Given the reliability and stability of GNU/Linux, on-call outsourced support is needed far less (and hence is far less costly) than for proprietary equivalents. In-house staff can be freed up to spend more time on constructive projects, as opposed to constantly putting out fires.

Firstly, Debian has a packaging system that makes it very simple to add new software whenever you need it, as well as update existing software with newer versions and security patches. Debian developers take packaging extremely seriously and work hard to make sure that packages are well tested and integrate well with other parts of the system. Want to install OpenOffice on your desktop? All you need to do is issue the command “apt-get openoffice.” Want to install the latest security patches? All you need to do is issue the commands “apt-get update ; apt-get upgrade” Debian takes care of all the software dependencies and installs the software for you, prompting you for important configuration decisions when necessary. There are even several GUI interfaces for the Debian package management system.

Debian packaging also has a sophisticated approach to managing stability versus being up-to-date. Debian has three main streams of software. You choose which you want to install on your hardware. *Stable*, which is released once a year is the closest one can get to stability nirvana in the IT world. Install Debian *Stable* on servers that you need to be running 24/7/365. You can install security patches for Debian *Stable* using the apt-get method, but you won't be getting the latest and greatest version of anything.

If you need/want more up-to-date versions of software, you can often find them in *Testing*, which is where the next generation of Debian is being tested. The bleeding edge stuff is in the Debian release known as *Unstable*. Actually, *Unstable* is a misnomer. I personally use *Unstable* on my desktop machines, because I *do* want the most feature-full version of desktop software. I've never had any problems with software not working or my system being messed up. In fact, for some of my software, I use experimental Debian packages that are not part of any "official" Debian release. Even with those, I have never had a problem. And in the worst case, you can easily un-install a package and re-install an older version.

Still, these who are in-the-know do not recommend *Unstable* or *Testing* for servers. For those occasions when you do need more up-to-date versions of certain software for your *Stable* server, you can always install from source. Remember, this is FOSS, so the source code is available and all modules can be compiled and linked to be compatible with the libraries installed on your system. Often, if it is an important piece of software, someone has back-ported it into a Debian binary package for *Stable*.

One final advantage of Debian is that it is available on a wide variety of hardware platforms, including Apple Macintosh hardware.

Unlike in the proprietary world, you are not limited to an off-the-shelf version of your operating system. Customizing the OS to fit your organization's needs is an option available to all, not just the very largest of companies. For this type of work, there are many good options. The founder of the Debian project has set up a company called Progeny (<http://progeny.com/>), which helps develop customized GNU/Linux deployments for sophisticated users. In the embedded Linux market, you might want to check out Opersys (<http://www.opersys.com/>). The ability to customize the OS unencumbered by licensing fees and restrictions has led major consumer electronics manufacturers to standardize on Linux as the embedded OS for consumer devices.

GNU/Linux File/Print Servers

GNU/Linux first found it's way into organizations as a safe and robust file/print server platform. Since this aspect of GNU/Linux is very well known and documented, there are only two points that need stressing here.

Besides the native NFS file servers, GNU/Linux is most often used with Samba (<http://us1.samba.org/samba/samba.html>) for serving files to Windows-based clients. Samba

3 has just been released. Many reviews have already declared it to provide far superior price/performance to its main competitor, Windows Server 2003. Anyone looking to upgrade their Windows Servers should take a good look at the GNU/Linux-Samba 3 alternative. Mac OS X also now provides excellent support for SMB, so Samba serves as an excellent product in mixed platform environments.

As for print serving, this was once a weak spot for GNU/Linux. But with the development of CUPS – Common UNIX Printing System (<http://cups.org/>) – GNU/Linux holds its own in this area as well.

GNU/Linux on the Desktop

It is common knowledge that GNU/Linux is suitable for servers and is not yet ready for the desktop. Just a couple of years ago, common knowledge said that GNU/Linux would remain a small/medium server option, and never be deployed in mission critical settings. And a couple years before that common knowledge said that GNU/Linux would never penetrate the corporate world, period, and would remain a hacker toy. So much for common knowledge.

If one does an inventory of the standard, day-to-day applications that employees use in any organization, they boil down to e-mail, browsing, and word processing, with the occasional spreadsheet thrown in. Microsoft Internet Explorer and Microsoft Office are the layers on top of the Windows OS that address these basic needs.

FOSS equivalents exist for all of these, which are superior in many ways to the Microsoft product. The Mozilla (<http://www.mozilla.org/>) family of products are excellent for e-mail, Web-browsing, and even calendaring, replacing Outlook functionality. OpenOffice.org (<http://www.openoffice.org/>) is quite a good replacement for Microsoft Word, Excel, and PowerPoint. Gnome Office (<http://www.gnome.org/gnome-office/>) is a collection of several standalone FOSS programs that are excellent, and address all the different office needs. For example, this document was written using Abiword (<http://abiword.com/>), which is a truly outstanding product.

Parts of Gnome Office are developed by Ximian (<http://www.ximian.com/>), which is a subsidiary of Novell. Ximian's Evolution product (<http://www.ximian.com/products/evolution/>) is a groupware replacement for Outlook that interoperates with Microsoft Exchange server.

One advantage to keep in mind is that many FOSS office products are cross-platform, existing in Windows, Mac, and Linux versions. As a first step, it is possible to migrate to a FOSS desktop without leaving Microsoft Windows. This is especially useful for smaller companies who, because of financial pressures, try to avoid buying multiple licenses of Microsoft Office products. Whatever one's position is on the ethics of "copying" software, given Microsoft's aggressive enforcement actions, using

unlicensed software is a risk best avoided. The best alternative is to move to FOSS equivalents. Licensing costs are free, and you usually get a more stable and useful product. Best of all, by using FOSS, your file formats finally are based on open standards that are documented and available to everyone; hence, you regain ownership of your data.

Once the migration to a FOSS office is complete, moving over to GNU/Linux as the underlying desktop OS is an easy step to take. The advantages are many:

- GNU/Linux provides a more stable, secure and robust environment.
- GNU/Linux runs well on older equipment, so hardware upgrades can be done less often.
- GNU/Linux has zero associated licensing costs, so you can avoid expensive upgrade costs to get newer functionality.
- GNU/Linux requires less support, so total cost of ownership is lower.
- GNU/Linux has excellent desktop alternatives that make the migration for Windows users relatively easy.

As noted earlier, major commercial software companies such as IBM, Sun Microsystems, and Novell are spearheading a drive to accelerate corporate adoption of GNU/Linux desktops.

There exist two big barriers to desktop migration. The first is file formats. Word, Excel, and PowerPoint files sometimes break their FOSS equivalents. This may cause problems when opening old documents or reading documents sent to you from customers or vendors. It is never a problem when you send documents to others, since all FOSS clients export files that are 100 percent compatible with their Microsoft equivalents.

One might argue that this is actually an argument supporting migration to FOSS. Why entrust sensitive organizational data to file formats only fully known to, and understood by, a monopolistic company? Moreover, there are many well-known security issues with these file formats, and Microsoft has done a very poor job of addressing them.

Over the past year, FOSS software has made huge strides in reverse engineering Microsoft file formats, so the problem of “broken files” is becoming far less of an issue. For both security and readability, many companies make use of PDF as a more robust file-exchange format. PDF works the same on all platforms, so it avoids the interchange issue completely. Finally, as XML formats grow in importance, interchange issues will become less problematic (all the FOSS desktop products use XML in their file formats).

The second problem in migration is dealing with legacy applications. In some companies, especially larger ones, sophisticated Word or Excel macros have been developed with which the FOSS equivalent software can't cope. Legacy documents that make use of these are problematic. Also, many companies have specialized Windows-based applications that have been developed in-house or bought from ISVs (independent software vendors).

One possible alternative for dealing with this problem is Wine (<http://www.winehq.com/>). As noted on the Web site, Wine is a FOSS implementation of the Windows API. By installing Wine on your desktop, you can make use of Windows legacy applications where necessary. There are commercially packaged versions of Wine that simplify its use. One excellent version is CodeWeaver (<http://www.codeweavers.com/>). One CodeWeaver product allows you to use Windows plug-ins in your Linux-based browser. Another product allows you to run Windows Office and many other Windows applications on your Linux desktop. CodeWeaver even has a server product that allows you to run many Windows applications on the server, which can be accessed on a user's Linux machine via a thin client.

Disney recently migrated its animation production to Linux workstations. Wine was an important part of this migration, since it allowed them to run Adobe Photoshop on Linux machines. Several other major movie studios are moving in the same direction.

VMWare (<http://www.vmware.com/>) offers another good option for running legacy Windows applications under Linux. Essentially, VMWare is a virtual computer that runs on top of a host operating system. Inside the virtual computer, you can install a full-fledged Windows OS with associated applications.

Over time, as GNU/Linux has gained popularity, more and more server applications have been ported to support it. Over the next couple of years, the same will begin to happen for desktop applications. But even now, there are many organizations, or employees within an organization, that can and should make the migration to GNU/Linux desktop.

Other FOSS OSes – The BSDs

GNU/Linux is not the only Open Source operating system available. A whole other family called Berkeley Software Distribution (BSD) exists. You can read the full history of the BSD at <http://www.openresources.com/documents/open-sources/node22.html>. Essentially BSD is an offshoot of AT&T's original Unix that was developed at Berkeley University, eventually under Defense Advanced Research Projects Agency (DARPA) support.

In 1991, Berkeley created a version of BSD that was free from AT&T Unix code, and so could be licensed without restrictions. In fact, the BSD license is far less restrictive

than the GPL, since a licensee can release the code modified or unmodified in source or binary form, with no accounting or royalties to Berkeley and no requirement to share modifications with anyone else. The only requirements are that the copyright notices in the source file be left intact and that products that incorporated the code indicate in their documentation that the product contains code from the University of California and its contributors.

By 1992, an Intel/386-based version of BSD was available. A group was formed to support this code, which is called NetBSD (<http://www.netbsd.org/>). NetBSD is targeted to highly technical users and runs on many hardware platforms.

Another group was formed known as FreeBSD (<http://www.freebsd.org/>), which originally was for PC architectures only, although over the years it has been ported to other architectures. FreeBSD has a GNU/Linux compatibility layer, so it can run GNU/Linux applications. But like the other BSD variants, it has a reputation of being even more stable, robust, and fast than GNU/Linux. Yahoo!, for example, runs its servers on FreeBSD.

Another interesting BSD-variant is OpenBSD (<http://openbsd.org/>) whose focus is security and integrated cryptography. Their Web site boasts that they had only one remote hole in the default install, in more than seven years. Of course, cynics note that OpenBSD is far less popular and widely deployed than other OSes, and so it's possible that its vaunted security is more a result of obscurity rather than any inherent superiority.

Commercial companies like the BSDs since they can modify them without restriction and package them as binary-only distributions. It was because of these licensing terms that Apple used BSD to create its own variant called Darwin (<http://developer.apple.com/darwin/>). Darwin is the core of MAC OS X. Apple's proprietary GUI runs on top of Darwin, but Darwin is also available as a standalone OS (<http://opendarwin.org/>).

Many of us remember that Apple wasted nearly 10 years and hundreds of millions of dollars trying to create a proprietary next generation OS. It is another tribute to the FOSS model that Apple finally got a modern, robust Macintosh OS by using FOSS code.

There is much cross-pollination between the GNU/Linux and BSD worlds, because they are both FOSS. An interesting development is the GNU/Linux compatibility for FreeBSD. A similar effort is underway for the Macintosh OS X. The Fink (<http://fink.sourceforge.net/>) project is porting many GNU/Linux applications to the Mac. Others, like Mozilla and OpenOffice.org, have been ported to OS X by the projects themselves. It is the Open Source nature of these projects that makes this cross-platform availability a relatively easy task.

FOSS and the Internet

We have mentioned several times how the explosive growth of FOSS has gone hand in hand with the growth of the Internet. The kind of collaborative development model that is essential to FOSS can best take place on the Internet. This is no coincidence, since the Internet was created as a medium for distributed collaborative communication. It is, therefore, not a surprise that much of the infrastructure software of the Internet is based on FOSS.

Basic Infrastructure

The vast majority of Internet Web servers run Apache (<http://apache.org/>). IBM's WebSphere product is based on Apache. The most common software for managing DNS is BIND (<http://www.isc.org/products/BIND/>). The most common software for managing e-mail is SendMail (<http://sendmail.org/>).

There are, however, alternative FOSS servers for these products. Qmail (<http://qmail.org/>) is the second most popular alternative to SendMail and is considered more reliable, safe, and robust. Another e-mail alternative is Exim (<http://www.exim.org>). A good IMAP server is Courier IMAP (<http://www.inter7.com/courierimap.html>). Squid (<http://www.squid-cache.org/>) is a well-known Web proxy cache.

Microsoft Exchange server bundles e-mail and collaboration functions in a relatively easy fashion. Recently, several FOSS alternatives have been announced. The most interesting of these is the Kolab (<http://kolab.kroupware.org/>) project, which was funded by the German government.

Another important FOSS Internet infrastructure product is OpenSSH (<http://www.openssh.org/>). This allows for secure encrypted login, copying, and FTP to remote servers. It is a vital tool for secure remote server management. Two well-known FOSS VPN products are OpenVPN (<http://openvpn.sourceforge.net/>) and Free S/WAN (<http://www.freeswan.org/>).

Web Scripting

Over the past decade, the most phenomenal growth area for the Internet has been the World Wide Web. Not only is the Apache server the most popular tool for serving Web pages, but FOSS products dominate the world of Web site production.

One of the most popular products for creating dynamic Web sites (i.e. Web sites where page content changes according to user input) is the scripting language PHP (<http://www.php.net/>). As its Web site states, "PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML." PHP is extremely popular – millions of Web sites use PHP. Yahoo! is

redoing it's whole Web site based on PHP. PHP is a scripting language specifically built for the Web, and is extremely easy to learn and use.

Perl (<http://www.perl.org/>) is another scripting language that was created in the late '80s, so it is not Web specific. Unlike PHP, it is not easy to learn nor is it easy to read Perl scripts. Nonetheless, it is extremely popular. It is also extensively used for scripting Web sites.

The most widely used modern programming language, at least according to the TIOBE Programming Community Index (<http://www.tiobe.com/tpci.htm>), is Java. Unfortunately, unlike PHP and Perl, Sun's version of Java is not considered FOSS because of Sun's restrictions on the code. There are many products, however, that try to create FOSS versions of Java. You can read about them at the Viva – Operation Free Java Site (<http://viva.sourceforge.net/>). Java is extensively used for scripting Web sites. One interesting FOSS Java product is the Apache add-on that is used to handle Java Web scripts, known as Tomcat (<http://jakarta.apache.org/tomcat/>).

Application Servers

Developers who are serious about developing sophisticated Web-based applications do not think highly of Web scripting languages. For one thing, there are security concerns regarding these languages. For another, developers often end up “re-inventing the wheel,” re-programming operations that are common across many different applications. Finally, sites based on these scripting languages don't necessarily scale well.

To deal with these issues, developers use what are called application servers. These servers are meant to be extremely robust, secure, and scalable. They provide sophisticated building-block services that allow developers to focus on application logic without having to implement infrastructure from scratch.

One very popular FOSS Java-based application-server is Jboss (<http://www.jboss.org/index.html>). Independent reviewers support the Web site's claim that JBoss competes with proprietary application servers such as IBM's WebSphere and BEA's WebLogic. JBoss also has a commercial company behind it that provides support for the product and guides its development.

Another popular Java-based FOSS application server is Enhydra (<http://www.jboss.org/index.html>). Enhydra is backed by a European FOSS consortium known as ObjectWeb (<http://www.objectweb.org/>). SUSE recently joined ObjectWeb, so Novell's recent purchase of SUSE will probably have interesting implications for the commercial popularity of Enhydra.

FOSS Protects Users from Commercial Failures

The history of Enhydra is illuminating regarding the advantages of FOSS. Enhydra was developed by a company called Lutris in the U.S. In 1999, Lutris released Enhydra as FOSS, although it later regretted it and returned to close-source development. The company itself failed, but the FOSS software it released had already created a community and continued to live on. This is a good example of how the FOSS model protects product users from the ups and downs of the commercial world. A good product will live on even if the company behind it fails. The users will not be left high and dry, and will not be economically punished by the failures of others.

Moving away from the Java world, another very popular FOSS application server is Zope (<http://zope.org/>). Like JBoss, Zope has a commercial company behind it that both supports Zope and guides its development. Zope is based on Python (more about this later). It is used by major organizations like NATO and CBS, and has many powerful add-ons and extensions (as do JBOSS and Enhydra).

Concluding this section, we note that any one of these application servers will provide a robust, secure Web application development platform competitive with any proprietary offering. They are excellent examples of how the FOSS ecosystem is moving up the software “feeding-chain,” away from basic infrastructure and into the world of middleware.

Content Management Systems (CMS)

Another important tool for developing sophisticated Web sites is a content management system (CMS). A CMS provides tools for users to modify the content and customize the organization of a Web site. All three of the FOSS application servers mentioned in the previous section have CMS add-ons.

One of the easiest and best FOSS CMS is Plone (<http://plone.org/>), which is an extension of the Zope platform. Plone allows developers to quickly set up a Web site to which authorized users can add content and modify layout with great ease. As the Plone Web site states, Plone is useful as “an intranet and extranet server, as a document publishing system, a portal server and as a groupware tool for collaboration between separately located entities.”

In particular, if one adds a Wiki (<http://plone.org/documentation/developer/>) and bug collector (<http://plone.org/collector>), Plone serves as an excellent collaborative communication environment for software development of any sort. Which leads us into our next section.

Software Development

Basic Tools

The previous section discussed FOSS in the context of one type of software development – Web applications. But FOSS software obviously provides tools for any kind of software development. From the very beginning, FOSS software was developed using FOSS tools. The first two tools that Richard Stallman developed for GNU is his FOSS version of Emacs (<http://www.gnu.org/software/emacs/>) and the GCC compiler (<http://gcc.gnu.org/>).

I had the pleasure of using Stallman's original Emacs on the DEC-20 computer when I was a student at Carnegie Mellon University. Emacs is far more than a text editor. It is actually a terminal-like environment that allows you to do just about anything your computer is capable of doing. Modern day Emacs even allows you to read your e-mail and browse the Web. Some hard core geeks who think GUIs are for wimps spend all their time on the computer within the Emacs environment.

In particular, Emacs serves as a basic Integrated Development Environment (IDE). Linux Journal had an excellent tutorial, "Emacs: the Free Software IDE" (<http://www.linuxjournal.com/article.php?sid=5765>), on using Emacs as an IDE with your favorite programming language. Because Emacs is extensible, you can add on IDE capabilities for any programming language. For example, the Java Development Environment (<http://jdee.sunsite.dk/>) for Emacs adds extensions that allow you to edit, debug, and document Java applications.

For people who find Emacs cumbersome or overwhelmingly feature-filled, a good FOSS alternative text editor is Vi (<http://www.vim.org/>). If Emacs is the Swiss army knife of text editors, Vi is the switchblade – fast, sharp, and efficient. A third popular alternative, which lies somewhere in between, is NEdit (<http://www.nedit.org/>).

The GCC compiler is an amazing tool that currently also compiles C, C++, Objective-C, Fortran, Java, and Ada. Perhaps it was the versatility of the GCC that convinced Linus Torvalds to base Linux on GNU. The GCC also now runs on a large number of hardware platforms.

GNU also contains the famous Unix utility. Make (<http://www.gnu.org/software/make/>). Make allows the developer to define all the rules and dependencies necessary for generating the completed application from its source. Make is programming language independent.

For documentation, the most widely used document processing system is LaTeX (<http://www.tug.org/>). There are many FOSS support tools for LaTeX. One of these worth noting is LyX (<http://www.lyx.org/>). In the words of the Web site, LyX "is an

advanced FOSS document processor that encourages an approach to writing based on the structure of your documents, not their appearance. LyX lets you concentrate on writing, leaving details of visual layout to the software.” LyX is based on LaTeX.

Advanced Tools

There are many FOSS advanced tools for software development, some general and others specific to the many programming languages that the FOSS world supports.

One of the most important tools for a software development project is software control and configuration systems i.e. a tool to manage and track changes and versions of software over the course of its development life. The most popular and widely used FOSS software for this purpose is Concurrent Version Systems – CVS (<http://www.cvshome.org/>). CVS has various limitations. In fact, Linus Torvalds has chosen a proprietary software control system to manage the development of the kernel. This is a source of great controversy, and illustrates the difference between the pragmatic Torvalds and the idealistic Stallman (who is incensed about this decision).

The FOSS world has risen to the challenge and a next-generation system is currently under development, known as Subversion (<http://subversion.tigris.org/>). Subversion is currently in alpha development stage, and already has almost all the features CVS does with many important extensions. It is already beginning to be used in some deployed software systems and promises to be an excellent alternative to CVS.

FOSS Paid Development

Subversion is an example of paid development in the FOSS world. During the late '90s stock market bubble, several companies were started by some stars of the FOSS world. One of these was CollabNet (<http://www.collab.net>), a company dedicated to providing tools and services based on FOSS development methods. One of its founders is Brian Behlendorf, a co-founder of the Apache Web server project. Software control and configuration is a key tool for software development, so it is not surprising that CollabNet would want to offer the best there is. Therefore, it pays for the salaries of several of the Subversion developers.

While some people may find this surprising, if one goes back to the economic model discussed above, it actually makes quite a bit of sense. Keep in mind that the distinction is not commercial vs. FOSS, but proprietary vs. FOSS. It makes good commercial sense for companies to invest in FOSS development. CollabNet invests a relatively small amount of money in R&D and gets the best minds out there supplementing their internal R&D department. CollabNet makes money by offering support, training, and consulting services around the FOSS tools it develops and packages. Subversion is a critical component. The fact that CollabNet might not actually sell Subversion as a packaged product is irrelevant. Moreover, there is nothing to prevent CollabNet from doing that, just like Red Hat and SUSE sell GNU/Linux, despite it being available totally for free.

While the bubble has burst, there remain many such examples of commercial support for FOSS projects, and it is a growing trend. It has been noted several times how IBM has made huge investment in FOSS development. But there are many far smaller software companies, such as CollabNet, who see the viable business model behind FOSS.

Another important advanced tool for software development is an IDE. Some programmers are minimalist and prefer working with just a text editor and a command line. For the next level up, there is Emacs. But for those who want a highly sophisticated IDE, there is Eclipse (<http://www.eclipse.org/>). Eclipse is a project that was originally developed in-house at IBM. Because of IBM's commitment to FOSS, it has been released as a FOSS product. While originally developed in and for Java, Eclipse has an extensible plug-in infrastructure and it now supports many other programming languages.

We mentioned Plone as a collaborative communication development environment. There are many stand-alone FOSS Wikis, bug-tracking tools, and other collaboration tools. There are also several Web-based environments that are integrated environments where FOSS collaboration takes place. Some of these are related to specific FOSS projects. For example, MozDev (<http://mozdev.org/>) is the home of projects related to Mozilla. Some are general purpose. The largest and most successful of these is SourceForge (<http://sourceforge.net/>). CollabNet has a similar environment called Tigris (<http://www.tigris.org>). These are excellent environments to learn about what is going on in the FOSS world, get involved in an existing FOSS project, or start one of your own.

FOSS Database Management Systems

Database management systems (DBMSs) are key tools for software application development, and are a critical component of most application infrastructure. There is quite a bit of work being done in FOSS world around DBMS.

There are four major FOSS database players.

1. MySQL (<http://www.mysql.com/>) calls itself “the most popular open source database server in the world.” If popularity means number of installations, then MySQL certainly wins. Starting out as a relatively simple data manager, MySQL has added full RDMS capability over the past few years, and now even has replication features.
2. PostgreSQL (<http://www.postgresql.org/>) has always been viewed as MySQL's big brother. It is the oldest FOSS database around, having been first conceived and developed at UC Berkeley in 1986. It has been a full-blown RDBMS from the beginning, unlike MySQL. Read “The History of PostgreSQL Open Source Development” at <http://developer.postgresql.org/pdf/history.pdf>, which is part of a short tutorial,

“PostgreSQL 7.2.1 Documentation” (<http://www.mediahost.org/docs/POSTGRESQL/>) about the system. Commercial support is provided by PostgreSQL Inc. (<http://www.postgresql.com/>).

3. Firebird (<http://firebird.sourceforge.net/>) is a FOSS spin-off from a commercial RDBMS, Interbase. Though its roots go back farther than PostgreSQL, it only became FOSS recently. In 1999, Borland felt that making Interbase open source would help it compete against Oracle, whose R&D budget could not be matched. Borland also was hoping to cash in at the stock market on the then current hype around FOSS. Borland later back-tracked, but once the code was out there, the project could continue to thrive without Borland’s support. Some of the original brains behind Interbase are still involved in Firebird. I chose Interbase as our RDBMS of choice when working on STATEMATE. It was a great product. I imagine it still is given the brainpower behind it. Commercial support is provided by IBPhoenix (<http://www.ibphoenix.com/>).
4. SAP DB, as the name implies, is a FOSSified version of SAP’s commercial RDBMS. SAP recently announced a partnership with MySQL, so it is unclear what its future as a separate product will be.

An interesting new development in FOSS database application development is ReKall. The Kompany (<http://www.thekompany.com/>) is an interesting organization that develops products based on KDE (<http://www.kde.org/>) – one of the two main FOSS desktop environments for GNU/Linux – and Trolltech’s GUI development API, QT (<http://www.trolltech.com/>) – the foundation for KDE.

The Kompany sponsors several interesting FOSS projects and has FOSS versions of several of their products. Recently, The Kompany announced the GPL release of ReKall, a RAD tool for building database applications, along the lines of Access. The name of the very first RAD database application development tool is long forgotten, but dBase started a revolution in database application development. dBase made such development extremely popular and, of course, Microsoft embraced, extended, and then ruled the market with Access.

Although many people look down on these tools as “database lite,” they serve an important function. They allow the actual application user to prototype the application. If it is useful and important, it can serve as the use case for the more robust, scalable, full-service app. So these types of tools are a critical component for agile software development (if used properly, of course). A good FOSS alternative has been sorely missing.

ReKall might actually serve as a good front-end development tool to robust database applications, since it hooks into RDBMS systems, and isn't just a "database lite" backend (as dBase and Access are).

ReKall's community site for the product (<http://www.rekallrevealed.org/index.php>) will hopefully have more information soon.

Middleware

We noted earlier in the discussion of application servers how FOSS is moving up the food chain in the software development ecosystem. The most explosive area of growth in the FOSS world is middleware, the software that mediates between infrastructure and user applications. We will only briefly touch on the vast world of FOSS middleware.

Java is the most popular middleware platform by far, and there are numerous FOSS projects related to the Java platform. We have covered quite a bit of Java resources in the earlier discussions. For a more in-depth look at non-Java middleware, the author has written another article, "Python – Language of Choice for EAI," which may be found at <http://www.fourm.info/MyArticles/pythonEAI>. Briefly, the article discusses how Python is a rapidly growing software development platform that matches Java in functionality, is wholly compatible with Microsoft's Java-competitor .NET, and is far easier to learn and use than either Java or .NET.

Twisted (<http://twistedmatrix.com/>) is a sophisticated, event-driven Python-based FOSS framework, which provides extremely powerful, scalable, and flexible enterprise application integration capabilities. Twisted allows developers to rapidly create sophisticated, safe, and robust network-based applications. It is an excellent example of how object oriented development tools can provide LEGO-like environments for building the most sophisticated applications. Critics in the proprietary world argue that true software innovation can only exist within a proprietary environment. Twisted is an excellent example that belies that statement.

Conclusion

Hopefully, this foray into the FOSS ecosystem gives the reader a sense of the vast amount of commercial-grade software available for just about any type of use, from software development to end-user applications. We encourage our readers to continue to explore this world. Someday, we hope to spot one of your contributions as well.

About the Author

Aron Trauring has been managing software projects for small and large companies around the world for over 25 years. He is currently the CEO of Zoteca (<http://www.zoteca.com/>), a company that offers FOSS IT consulting services. Trauring is on the Board of Directors of the New York Software Industry Association and is the leader of its FOSS Special Interest Group. You can also find more of his writings and opinions about FOSS at: <http://fourm.info>.

If you have any questions or comments about this white paper, feel free to e-mail us at: whitepapers@infotech.com



Practical solutions for IT managers of mid-sized enterprises

Info-Tech Research Group is a professional services firm dedicated to providing premium research and objective advice to IT managers of mid-sized enterprises. Currently, we serve more than 18,000 clients, primarily across North America and the U.K.

Our purpose is to provide practical and thorough solutions that enable IT managers to bridge the gap between technology and business. Our Web publications, guides & reports, methodologies, and consulting services all help IT professionals to keep current, to effectively manage people and technology, and to achieve professional success.

Products and Services

To serve our growing client base, Info-Tech offers a broad range of products and services. Our two **Web publications** reach more than 18,000 clients worldwide. **Info-Tech Advisor** focuses on providing critical and practical information to IT professionals, allowing them to successfully tackle their daily challenges. **McLean Report** is geared towards the strategic use of technology. It provides a comprehensive look at the interplay between IT innovation and business strategy.

Guides & Reports is another key product area offered by Info-Tech. From benchmarking data collected from our client base, and thorough research conducted on over 800 secondary sources, our analysts glean the critical findings and patterns to make insightful and relevant recommendations: the exact information our clients need to take the next step in their decision-making process.

To serve the unique needs of our mid-sized market, Info-Tech offers **Consulting Methodologies** that we have developed and tested through our past consulting engagements. These step-by-step methodologies have been packaged into user-friendly formats to facilitate clear understanding and independent implementation.

Consulting Services are also available from Info-Tech. Rather than the traditional hourly rate pricing format, we have developed several high-value programs involving clear deliverables and fixed pricing. This ensures our clients will have clear expectations of the results to be delivered, coupled with no hourly rate surprises.

Note: All Web links in this document were checked for accuracy and functionality at the time of publication. We cannot, however, guarantee that referenced Web sites will not change the location or contents of linked materials, and will not be held responsible for such changes.

To find out more - visit www.infotech.com

© Info-Tech Research Group, 2003.