

Standards Today

A Journal of News, Ideas and Analysis

A publication of
**CONSORTIUM
INFO.ORG**
GesmerUpdegrove LLP

August – September 2009

Vol. VIII, No. 5

OPEN SOURCE AND GOVERNMENT

EDITOR'S NOTE: [What is this thing you call "FOSS?"](#) 3
Free and open source software isn't the easiest thing to understand. But it's worth making the effort.

EDITORIAL: [FOSS and Government Procurement:
Dispersing the Fog of Lobbying War](#) 5
Some of the best software available is open source, but non-proprietary software has enemies as well as friends. If the Obama Administration expects to achieve its ambitious, technology-based policy goals, it would be wise to publicly declare its support for FOSS.

FEATURE ARTICLE: [A Concise Introduction to Free and
Open Source Software](#) 8
Software that could be freely edited existed long before proprietary programs became the norm - but then it largely disappeared. When source-available, "free and open software" (FOSS) reemerged in the marketplace, it did so in a manner that was novel from both a social as well as a legal perspective. Today, it is an increasingly important part of the information technology landscape. In this article, I provide an overview of the history, legalities, social theory and commercial impact of the FOSS phenomenon, as well as some thoughts about its future.

STANDARDS BLOG: [Nurturing the Flower of FOSS at
the CodePlex Foundation](#) 48
Successful FOSS projects are invariably based upon a delicate balance of power between individual developers and corporate interests – a mutually beneficial relationship that does not always develop. In two recent blog entries, I gave my views on how that feat can best be achieved at the open software support foundation just launched by Microsoft.

Standards Today is a free bi-monthly electronic Journal sponsored by the Boston law firm of Gesmer Updegrove LLP. The current issue of **Standards Today** and a subscription form may be found at www.consortiuminfo.org/bulletins. Questions or comments about the articles in this issue or about ConsortiumInfo.org may be directed to Andrew Updegrove at updegrove@consortiuminfo.org, or by telephone at 617/350-7800. To learn more about Gesmer Updegrove's standards and open source practice, visit http://www.gesmer.com/practice_areas/consortium.php, or contact Andrew Updegrove.

© 2009 Andrew Updegrove. All rights reserved.

CONSIDER THIS:

The Constantine Code and
the *Missing Standard!*..... 64

Dan Brown is renowned for spinning tales that weave together ancient events and contemporary intrigue with potentially dire modern effects. I guess he does OK, considering what he has to work with. But next time, he should look into standards (*what the hell?*)

EDITOR'S NOTE:

What is this Thing you Call “FOSS?”

Free and Open Source Software (“FOSS”) is not the easiest concept to understand, even for technology folks when they first happen upon it. For legislators, it’s even harder to get their brains around something (like Linux) that isn’t geographically “from” anywhere, is created in real time by a global network of volunteers, isn’t owned by any one entity, is available for free – and has been compared by a certain prominent technology CEO to a manifestation of communist principles. As you can imagine, all that unfamiliarity doesn’t help when FOSS competes with proprietary software in the government procurement process.

But because of the ever increasing number of superior FOSS solutions that are available, it is essential that governments – national, state and local – give equal consideration to FOSS alternatives to proprietary software, if these public servants wish to responsibly spend the tax dollars of those that put them in office.

It is for this reason that I am dedicating this fifth and final issue in a series focusing on the standards-related needs of the Obama administration’s technology-based policies to the phenomenon of FOSS. My hope is that this issue will provide a useful reference for those legislators, and their staff, who need to become familiar with FOSS, perhaps for the first time.

I begin that task with my **Editorial**, which highlights the hazards of ignoring FOSS in situations where it provides the best tool to perform a given job. I also call on the Obama Administration to publicly declare its commitment to giving equal priority to FOSS alternatives in public procurement, to ensure that government CIOs have the support they deserve when choosing FOSS over proprietary solutions in the highly competitive, often rough and tumble, procurement process.

The **Feature Article** in this issue is intended to support government CIOs in a different way, by providing those on the Hill with a comprehensive overview of FOSS: what it is, where it comes from, and how to use it, all in one place. My hope is that this concise but broad reaching survey will prove to be a useful guide for decision makers that may have found it challenging in the past to understand what FOSS is, and why they should care.

I follow with a “twofer” **Standards Blog** selection, occasioned by Microsoft’s launch of a new non-profit organization (the CodePlex Foundation) intended to facilitate the interaction of commercial companies and individual developers in FOSS projects. The purpose of this two part series was to explain the types of governance and development mechanisms that the founders of a FOSS project



[Tux, the Linux Mascot](#), by
Larry Ewing

need to put in place in order to successfully attract the kinds of participants, and enable the types of activities, essential to make such an enterprise succeed.

My **Consider This** piece closes on a different note, sparked by the release of Dan Brown's latest blockbuster symbologist mystery. In it, I show that when it comes to ancient religious intrigues with long-lasting effects, Brown's imagination comes in second to facts – and standards.

As always, I hope you enjoy this issue. But either way, it's always great to hear what you think. Let me know, why don't you?

Andrew Updegrove
Editor and Publisher
2005 ANSI President's
Award for Journalism

The complete series of Consortium Standards Bulletins can be accessed on-line at <http://www.consortiuminfo.org/bulletins/> It can also be found in libraries around the world as part of the EBSCO Publishing bibliographic and research databases

Sign up for a free subscription to **Standards Today** at

<http://www.consortiuminfo.org/subscribe/2.php?addentry=1>

EDITORIAL:

FOSS and Procurement: Dispersing the Fog of Lobbying War

Andrew Updegrove

People in Congress have it tough.

They're expected to deal with every new topic that comes down the pike, from regulating securitized credit swaps to beefing up cybersecurity, whether they've had any previous experience in the area. Of course, there's never a shortage of people who want to educate them, but the "educators" with the greatest access are likely to be lobbyists. And when one paid advocate is promoting one action, political physics dictates that another highly paid individual in somebody else's pocket will be promoting an equal and opposite action. Soon, all potential solutions become obscured by a fog of business propaganda.

What's a poor legislator (and her staff) to do?

Good question. There's been plenty of fog on Capitol Hill about free and open source software (FOSS) for a decade now, and that's hardly surprising. In the beginning, most big software companies were a'gin it, and any government agency CIO allowing a useful bit of FOSS to find a home on the servers she supervised was not likely to advertise that fact.

It's time for the Obama Administration to declare its public support for FOSS procurement by the federal agencies

Eventually, some major vendors (like IBM) began including FOSS programs in the systems they were promoting to government customers. But that only made things worse, because other vendors (like Microsoft) went actively on the offensive, initially comparing the theory and appeal of FOSS to that of communism.

With hordes of lobbyists deployed on the ground in Washington and in state capitals throughout the nation, it was soon a brave, and perhaps foolhardy, public IT manager who would dare to take a public stand in defense of FOSS.

And yet, by the turn of the millennium, FOSS was already infiltrating federal and state government systems everywhere. And it was spreading rapidly. When MITRE conducted a two week email survey of FOSS use within the U.S. Department of Defense in 2002, IT manager respondents reported that they were administering 251 FOSS installations that included a total of 115 different FOSS applications. Moreover, MITRE noted that the use of certain FOSS solutions had passed from elective to essential, giving examples that resonate even more forcefully today:

The main conclusion of the analysis was that FOSS software plays a more critical role in the DoD than has generally been recognized....One unexpected result was the degree to which Security depends on FOSS. Banning FOSS would remove certain types of infrastructure components (e.g., OpenBSD) that currently help support network security. It would also limit DoD access to - and overall expertise in - the use of powerful FOSS analysis and detection applications that hostile groups could use to help stage cyberattacks. Finally, it would remove the demonstrated ability of FOSS applications to be updated rapidly in response to new types of cyberattack. Taken together, these factors imply that banning FOSS would have immediate, broad, and strongly negative impacts on the ability of many sensitive and security-focused DoD groups to defend against cyberattacks.*

In the years since MITRE performed that study, FOSS usage has spread hugely on all types of government, academic, business and personal systems. In some areas, such as Web server installations, the use of FOSS software greatly exceeds that of all proprietary alternatives combined. Indeed, the use of FOSS is so pervasive that even Microsoft has accepted the reality that its enterprise customers will continue to

to maintain mixed proprietary and FOSS systems, and will expect Microsoft to help them maximize the efficiency of these systems as well.

Only by giving equal opportunity to FOSS can the Obama Administration ensure that, when it comes to getting the best deal for the American public, the best software will win

The result is that more and more of the best software developers participate in FOSS projects, whether on their own initiative, or at the behest of their employers. In consequence, some of the best architected, most effectively updated, and lowest total cost of ownership software to be found anywhere is now FOSS. The number of examples of such software continues to grow.

This is good news for the Obama administration, because the President cannot make good on his promises of openness, achieve his technology-dependent policy agenda, reduce the national budget deficit, or protect the nation against cybersecurity threats unless procurement officers actively embrace and utilize the best software available in every case, whether it be proprietary or FOSS.

Increasingly, the best software tool for a given job is likely to be the FOSS alternative. Moreover, choosing a FOSS product – and especially one that fully implements open standards – will better protect the procuring agency from vendor lock-in. It will also facilitate an ongoing high degree of competitive bidding for supporting services throughout the useful lifetime of a product, and provide

* United States Department of Defense, "[Use of Free and Open Source Software \("FOSS"\) in the Department of Defense](http://www.isd.mel.nist.gov/projects/rtlinux/dod-mitre-report.pdf)," Version 1.2.04 (MITRE, January 2, 2003), at: <http://www.isd.mel.nist.gov/projects/rtlinux/dod-mitre-report.pdf>

government IT managers with real-time time access to security and other bug fixes as well.

But will procurement officers always make the best and most informed choice between proprietary software and FOSS alternatives? Too often, government IT managers are subject to vendor pressures, and some (as in Massachusetts) have found it necessary to resign in the face of vendor retaliation. When it comes to FOSS, government CIOs would welcome some long-overdue fog cutting on the Hill. They deserve to know that someone has their back when they make the right procurement decisions for the benefit of the American people.

It's time for the Obama Administration to publicly state that it whole-heartedly supports FOSS procurement by the federal agencies. Not in preference to proprietary software, but on an equal basis. Only by doing so can it ensure that when it comes to getting the best deal for the American public, the best software will win.

Copyright 2009 Andrew Updegrove

Sign up for a free subscription to **Standards Today** at

<http://www.consortiuminfo.org/subscribe/2.php?addentry=1>

FEATURE ARTICLE:

A Concise Introduction to Free and Open Source Software

[Andrew Updegrove](#)¹

Abstract: *In the early days of information technology (IT), computers were delivered with operating systems and basic application software already installed, without additional cost, and in editable (source code) form. But as software emerged as a stand-alone product, the independent software vendors (ISVs) that were launched to take advantage of this commercial opportunity no longer delivered source code, in order to prevent competitors from gaining access to their trade secrets. The practice also had the (intended) result that computer users became dependent on their ISVs for support and upgrades. Due to the increasingly substantial investments computer users made in application software, they also became "locked in" to their hardware and software vendors' products, because of the high cost of abandoning, or reconfiguring, their existing application software to run on the proprietary operating system of a new vendor. In response, a movement in support of "free software" (i.e., programs accompanied both by source code as well as the legal right to modify, share and distribute that code) emerged in the mid 1980s. The early proponents of free software regarded the right to share source code as an essential freedom, but a later faction focused only on the practical advantages of freely sharable code, which they called "open source." Concurrently, the Internet enabled a highly distributed model of software development to become pervasive, based upon voluntary code contributions and globally collaborative efforts. The combined force of these developments resulted in the rapid proliferation of "free and open source software" (FOSS) development projects that have created many "best of breed" operating system and application software products, such that the economic importance of FOSS has now become very substantial. In this article, I trace the origins and theories of the free software and open source movements, the complicated legal implications of FOSS development and use, and the supporting infrastructural ecosystem that has grown up to support this increasingly vital component of our modern, IT based society.*

Introduction: If you have at least a passing acquaintance with information technology (IT), you will likely recognize the phrase "open source software" (OSS). If you're familiarity with IT is more detailed, you will also have heard the phrase "free and open source software" (FOSS), and perhaps the somewhat puzzling word combination, "free, libre open source software" (FLOSS). Unless you have made an effort to dig deeper into what lies behind these acronyms, however, you may be at

¹ **Disclosure:** the author and his law firm have acted as legal counsel to a number of entities mentioned in this article, including the Free Standards Group, the Linux Foundation, United Linux, and the X Window Consortium. Note: all Web pages cited in this article were last accessed on October 19, 2009.

a loss to give a coherent definition of any of these phrases, much less a differentiated explanation of each.

The very existence of so many names for what would appear to be similar, or at least related, concepts suggests that there may be more to be learned than immediately meets the eye. And indeed this is the case. The evolution and current state of FOSS² includes elements of political philosophy, revolutionary zeal, technical development methodologies, traditional as well as radical legal theories, and cold, hard business pragmatism. Needless to say, such a rich stew of attributes is likely to present something of a challenge to anyone interested in gaining a quick understanding of exactly what this new IT phenomenon is all about – and why, in some ways, it is not a new phenomenon at all.

The reasons for investing the time to gain a better understanding of FOSS are several. From a sociological point of view, the FLOSS movement (as compared to the development of OSS) is part of a broader, socio-political movement, energized in part by the ability of the Internet to enable the sharing of information and the active collaboration of people on a global basis. That movement questions the utility and fairness of many traditional copyright

The FLOSS movement is part of a broader, socio-political movement, energized in part by the ability of the Internet to enable the sharing of information and the active collaboration of people on a global basis

and patent-based legal restrictions, and seeks to liberate information for the benefit of all. In the case of FLOSS, it also articulates a set of ethical rules intended not only to foster free access, but also to inspire – and in some cases require – those that benefit from such access to contribute their own modifications and additions to FLOSS back to the common weal as well.

From an economic point of view, the FOSS development model is reordering the business realities of software development in multiple ways: for a developer, the per-business costs of development of a given piece of software can be radically reduced by participating in a development project in which many others contribute their efforts as well; for an end user, access to the source code of a FOSS product grants independence from a proprietary vendor, since the end user can adapt the code itself, or put development work out for competitive bidding; for software vendors, profit opportunities move towards value added services and away from product design; and from a marketplace perspective, the FOSS model presents a disruptive force that offers opportunities for both existing, as well as new, businesses to attack the dominance of entrenched market participants whose advantages rest on the proprietary development and sales model.

And finally, from an overall business perspective, FOSS has become so thoroughly embedded in the reality of the modern marketplace that effective IT procurement

² In this article, I use the word FOSS to mean software made available under a license that complies with the “free software” OR the “open source” definitions that are discussed further below; FLOSS to refer to software released under a license that satisfies the free software definition (and in all likelihood, but incidentally, the open source definition as well); and “OSS” to refer to software made available under a license that meets the open source, but not the free software, definition.

and management is becoming increasingly difficult absent a working knowledge of what FOSS is all about. Active participants in the development and use of FOSS products additionally need to know how FOSS can be expected to evolve in the future, and how the legalities of FOSS apply to anyone that participates in the development of FOSS, uses a FOSS product, or embeds any FOSS code in their own products for resale.

In this article, I will provide an overview of the history of FOSS and its champions, the major philosophical differences that divide FOSS from OSS developers, the multiple licenses under which FOSS is made available, and the principle non-profit institutions that support and promote FOSS. I will conclude with a brief bibliography of primary FOSS sources for those that wish to learn more than this necessarily superficial review can hope to provide regarding such a rich and complex topic.

I FOSS: The Basics

What exactly does someone mean when they use the acronym “FOSS” or the phrase “open source?”

What it is (and what it isn’t): The answer is not only “it depends,” but that it depends a lot more than one might think. At one extreme, it may mean very little, because the person using the phrase has only a very general idea of what the words mean. At the other, it may imply quite a broad spectrum of information, covering topics as varied as legal rights and obligations, affiliation with social movements, and the modes of development. In other words, the words open source, and in particular, “free and open source software,” may mean many things at once, as dictated by the context and the knowledge of the individual that uses them.

At the most basic level, OSS is simply a piece of software for which both machine readable (object) and human readable (source) code is supplied to the user. And sometimes, this is all that the words “open source” do connote, as when a single developer creates a piece of code and then posts it to the Internet at a public site with few, or no, restrictions on its reuse.

A popularly used FOSS program, however, is likely to have additional attributes that differentiate it from proprietary software. Most likely, it will have been developed, and will currently be maintained, at a public Web site that allows any interested programmer to sign up and offer to help, whether by pointing out bugs and suggesting ways to fix them, by actively participating in code development, or by helping document the ongoing work of others as it happens.

Moreover, the software that a given project makes available may be not just a single program, but a package of carefully integrated FOSS software intended for a particular purpose. The project making the package available may have actually developed only one or a few of the components, with the balance coming from other sources.

The project in question may have been started by a single individual, or by a group of individuals, or it might have been launched when a proprietary vendor released the object and source code to a product that it had developed, concluding that it would gain greater benefit as a result of doing so (e.g., by having continuing access to the same code at a lower cost, due to the labor contributed by non-employees, or by selling support services to the users that download the program for free).

The project may be supported or hosted in a variety of ways as well. Perhaps its existence is wholly virtual, with the code residing on a bank of servers hosting thousands of other FOSS projects, some inactive, while others hum with the activity of engineers spread around the globe, few, if any, of whom have ever met – or perhaps even know each others real-world names. Or it may be

supported primarily by a single vendor, with many of its most active participants working for that employer, and under the same roof.

What FOSS is not is an involuntary infringement on any developer's rights, a second best alternative to proprietary code, or a security risk to the enterprise

Legally, the term may imply that anyone can download the code and do whatever they want with it, so long as they do not try to sue the developer for any flaw in the code or infringement of the rights of any third party. Or it may indicate that anyone that changes the code and sells the modified version must contribute its own code back to the project as a matter of ethics and morality, as well as in response to a legal obligation.

What FOSS is *not* is an involuntary infringement on any developer's rights, a second best alternative to proprietary code, or a security risk to the enterprise.

And it certainly isn't a passing fad. FOSS is here to stay.

The value proposition: The value of "free software" for the customer sounds obvious. But for the developer, the appeal seems far less intuitive. In each case, though, the value is very real, and can be found in a variety of ways, some of which are not immediately obvious. Briefly stated, they are as follows:

For the customer: While individuals typically incur a one-time cost to acquire software and are then on their own, commercial customers are likely to make a more substantial investment in additional services, such as purchasing training for their employees to learn how to use the new software, and also ongoing "support" services (i.e., ensuring that there is someone at the end of the phone if problems are encountered installing, integrating, or operating the software on complex enterprise systems), as well as "maintenance" fees to ensure that they will get updates (e.g., bug fixes and improvements) after the software has been installed. They may also need to pay for hardware upgrades in order to be able to run the new software, and pay consultants and other service providers to plan and complete the upgrade.

The aggregate of all of these fees is the "total cost of ownership" of a given software package, and the total can be substantial, even where the software itself

can be obtained without charge. Similarly, while some FOSS distributions may be free (e.g., the OpenOffice office suite), a customer may decide to pay for a more fully featured version of the same software with ongoing, included service (e.g., Sun Microsystems's StarOffice variant of the same basic software).

Will the final cost of a FOSS product therefore invariably be cheaper than the proprietary alternative? As can be imagined, the point is one that has been hotly debated, and has been the subject of a variety of studies, only some of which have been commissioned by parties with no vested interest in reaching one conclusion or



the other. There is agreement, however, on a number of empirical, as well as experiential benefits to using FOSS over proprietary products:

- ✓ **Access to code:** When a user installs proprietary software, they become entirely dependent on the vendor for its quality, improvement and performance, because the customer has neither the technical means (access to source code) nor the right (legal permission) to alter the code. If the customer needs new or different features, the vendor may or may not be willing to customize the program (either at all or at a price the customer is willing to pay), and if the vendor discontinues support for the product, or goes out of business, the customer is stranded.³ In contrast, a customer with a FOSS alternative has the ability as well as the legal right to change the code any time that it wants to. It can also hire anyone it wishes to help it change or maintain the code, and if the project that created the code goes dormant, it may be disappointed, but it will not be stranded.

By analogy, the unlucky user of a proprietary product is like someone that rents a house from a distant and unresponsive landlord that may be erratic in its maintenance and someday may even go bankrupt, while the user of a FOSS product is like the owner of her own home, and is free to renovate whenever she wants, doing the work herself or contracting the carpentry and painting out to the lowest bidder.

- ✓ **Freedom from lock in:** While open standards increasingly give customers protection from "lock in" (i.e., dependency on a single vendor, and the certainty of significant switching costs if they wish to change vendors), changing from one product to another can still be difficult and expensive in many situations. In the case of systems based on Linux, the increasingly popular FOSS operating system (OS), there are currently over 600 independent "distributions," all based on the same core software (the Linux kernel). While application software will not always automatically run across all versions, all of the major distributions certify their products to the Linux Standards Base (LSB), a set of standards currently maintained by the Linux

³ The standard defensive tactic for a licensee is to negotiate the placement in escrow of a periodically updated copy of the source code to the software, accessible only if the licensor fails to support the customer as promised. However, licensors are very unwilling to make such a term broadly available, and only the largest customers, and governments, are therefore able to obtain such a term with any frequency.

Foundation, in order to permit application software to run more interoperably across all compliant distributions.⁴

- ✓ **Release cycles and bug fixes:** Well run FOSS projects are in constant motion, upgrading and fixing bugs in real time. Customers can access this work on a far more frequent basis than users of proprietary products, who must wait until the vendor decides to incur the costs of making a minor or major release. Because the source code to FOSS is available to the customer, popular FOSS software also generates a flood of bug reports and suggested fixes, which are evaluated and implemented as appropriate on a constant basis.

Or, as stated in what is often referred to as "[Linus's Law](#)" (as in Linus Torvalds, the originator and ongoing leader of Linux development): "Given enough eyeballs, all bugs are shallow." In contrast, proprietary vendors only receive complaints from customers, and then must seek to replicate and diagnose the problem before they can fix it.

- ✓ **Security:** While it may seem counterintuitive that code visible to anyone anywhere would be safer to use, popular FOSS programs are acknowledged to be more secure, largely for the same reasons just stated: because anyone can see the code, anyone can track down the source of a vulnerability, let project managers know of the cause of concern, and/or propose a fix herself. As a result, security issues can typically be identified, fixed, and propagated to all users faster than flaws in proprietary code.⁵ As a result, FOSS is increasingly being used by defense, financial and other types of users where security concerns are greatest.⁶

For the developer: In the first instance, it is important to note that FOSS is created by individuals, rather than by companies. Individuals participate for many reasons, including enjoyment, challenge, gaining status within the project community, and gaining valuable job skills that enhance marketability and compensation potential.

Points of origin: Those first becoming acquainted with FOSS are often puzzled by the fact that there may be no physical "there" there, in the sense of a central development facility. This is hardly surprising, because in many cases there is no legal entity that owns, or that is responsible for creating or maintaining the code

⁴ The LSB was originally developed and maintained by the Free Standards Group. FSG was merged into Open Source Development Labs (OSDL) in early 2007, with the combined entity changing its name to the Linux Foundation. LF continues to maintain the LSB today.

⁵ Indeed, popular FOSS projects can, and are, scanned on a regular basis and ranked for security and security improvements. In the most recent report (2009) in a series of studies originally launched at the request of the United States Department of Homeland Security, security scanning tools vendor Coverity announced that the incidence of flaws detected over a three year scan of 11 billion lines of code from 280 most-used open source projects (e.g., Firefox, Linux, PHP, Ruby and Samba) over three years had declined by 16%. See, [Coverity Scan Open Source Report \(2009\)](http://scan.coverity.com/report/Coverity_White_Paper-Scan_Open_Source_Report_2009.pdf) at: http://scan.coverity.com/report/Coverity_White_Paper-Scan_Open_Source_Report_2009.pdf

⁶ A survey of Department of Defense IT managers conducted by MITRE in 2002 concluded that FOSS was essential to maintaining secure systems. See, United States Department of Defense, "[Use of Free and Open Source Software \("FOSS"\) in the Department of Defense](#)," Version 1.2.04 (MITRE, January 2, 2003), at: <http://www.isd.mel.nist.gov/projects/rtlinux/dod-mitre-report.pdf>

(Linux, which is created by a global network of thousands of individual developers, is a prime example). Instead, the code may simply be hosted, often for free, at a server farm maintained for that purpose by an organization such as SourceForge, which also provides a variety of supporting tools and services.

Other projects are supported by non-profit foundations formed for that purpose (e.g., the Mozilla Foundation, which supports the Firefox and Mozilla Web browsers). Only a small minority of FOSS projects are supported by for-profit corporations, such as Red Hat or Novell, two commercial companies that profit by offering unique Linux-kernel based distributions, plus support services, based upon the project that it funds (Fedora and SuSE, respectively).

While selling services is a popular model for profiting on FOSS, it is not the only one. Simply sharing the development costs of software with other companies needing the same software tools lowers the overhead per vendor. Again, just as collaboratively developed open standards permit competitors to vie with each other in other ways (e.g., by developing and selling proprietary features and services offered above the level of standardization), FOSS can enable entirely new and competitive business opportunities. A current example can be found in the mobile device (e.g., smartphones, PDAs and netbooks) marketplace, where multiple Linux-based operating systems (e.g., LiMO, Android, and Moblin) have been developed in order to fuel new opportunities for the companies that funded their initial development (multiple companies, in the case of LiMO, Google, in the case of Android, and Intel, in the case of Moblin).



The reality just described is of recent vintage, however, and many of the realizations outlined above were unknown to most current users as little as a decade ago. Given the breadth of uptake of FOSS in such a short period of time, how did such a counterintuitive transformation take place?

II A Brief History of FOSS

For reasons that will become clear, it is impossible to fully comprehend what FOSS is all about without understanding how it came about. Indeed, the early history of open source is as much a tale of techno-revolutionary vision as it is of computer coding. The details of this still-evolving process can (and already have) filled books, and thus the following overview will necessarily be both selective and brief.⁷

The rise of proprietary software. For most non-IT professionals, computer software became “real” around the time of the first sales of the first IBM PC in 1982. Following the initial success of Apple, and then IBM personal computers, almost all commonly used desktop software has been proprietary, meaning that the

⁷ For a more detailed account of the birth of FOSS, as told by someone who participated in that process, see: Salus, Peter H., *The Daemon, the Gnu and the Penguin* (Reed Media Services – 2009). Another excellent, contemporaneous account is Glyn Moody's *Rebel Code* (Basic Books – 2001).

user must purchase a copy of the software from the company that developed it. Having done so, a home computer user is dependent upon the same developer for any future upgrading of the product that may be needed to ensure that the owner can still make use of it. If the vendor goes out of business, or discontinues support for the software, the user will find that over time the product is likely to become useless, either because it will not run on a new computer or game console, or because its output (e.g., a document file or spreadsheet) can no longer be exchanged with another computer user.

At the platform level, most computer users have spent their entire lives in the locked-in “Win-Tel” world that sprang from the marriage of Microsoft operating systems with Intel processors. And in the realm of application software, most of the same users have lived (when it came to office productivity tools) in the same convenient, but constrained world of Microsoft Office. Convenient, because

Like a blueprint, access to source code provides a means by which a software architect can readily understand and change the design of a program

everyone else also used Office, and text documents and spreadsheets could therefore be easily exchanged among other Office users. But constrained, because once a user entered the world of Office, it was very difficult to leave it (until quite recently).

Business users, including those with their own IT staffs, find themselves in the same position when they purchase proprietary software, for two reasons. First, the legal basis upon which businesses (like home users) obtain software is not a sale at all. Rather, the user obtains the rights to use the product under a “license.” The transaction is superficially similar (i.e., the customer sends a check, and the vendor sends the software, and typically retains no right to demand its return). But the legal differences are nonetheless significant, because a license permits the software vendor to impose restrictions on use that would be impossible to enforce in the case of transaction structured as a sale – even the right to forbid the customer to make further use of the software if she violates the license terms.

Second, the product is provided only as “object code,” meaning in a form that a computer can readily read and execute, but which a human being can only interpret with great effort through a process known as “reverse engineering.” Only customers with a great deal of clout are able to require delivery of a copy of the software in “source code” form as well, which allows a developer to understand and change the product as they wish. By way of a very rough analogy, source code is to object code what a blueprint is to a finished building. Like a blueprint, access to source code provides a means by which a software architect can readily understand and change the design of a program. Source code also provides a roadmap permitting software engineers (and automated computer programs) to execute those changes.

Curiously, limiting software deliveries to object code only is a recent business practice rather than the original means by which software was made available to customers. In the early days of computers, vendors focused on hardware design, differentiating themselves from their competitors through hardware features and

performance. The operating system for a computer, and often other software as well, was provided by the hardware vendor without additional charge, and in source as well as object code form so that the customer's own software engineers could build their own custom software to run on top of the vendor's OS.

This was good for the vendor for two reasons. First, because a robust industry of independent software vendors, or "ISVs," had not yet developed, and a computer is useless without software. As importantly, providing software in source code form encouraged a customer to change and build upon that software. Over time, this meant that a typical customer of a mainframe (and later a mini) computer would make a very substantial investment in the software it relied upon to run the core functions of its business (e.g., payroll, inventory, fulfillment, and so on). Since the hardware vendor owned the rights to the operating system upon which (typically only) its own computers ran, its customers became "locked in" by the very substantial costs required to adapt (or "port") the custom software it had developed to the operating system of any other vendor.

Several developments in the marketplace changed this initial, easy-going approach to source code. One was a fundamental change in the market for operating systems, beginning with the decision by IBM to license from Microsoft, rather than internally develop or purchase, an OS for the line of "personal computers" that it decided to sell. It also opted to buy, rather than design, the computer processors that would power its new PC line – even though it already had an existing processor that would have been suitable for this use. That fateful decision allowed Microsoft, the owner of the OS selected by IBM, and Intel, the developer of the chip design, to license the same products to others. The result was the rapid development of the PC "clone" computer, an often cheaper unit that could be sold in direct competition with IBM – and was, in vast quantities.⁸



IBM introduces the PC
August 21, 1981

Microsoft made another crucial decision with broad and lasting impact on the development of software. Before it secured its OS contract with IBM, Microsoft had been a developer of compact versions of the BASIC programming language software for the Altair, Commodore, Apple II, Radio Shack TW-80 and other early computers with limited processing power. After Microsoft entered the OS market, it continued to launch application software, and preferentially for use on IBM and IBM clone PCs. Microsoft did give access to enough data about its DOS (and later Windows) OSs to other ISVs to allow them to develop their own application

⁸ Microsoft, as it turned out, also lacked an operating system when the world's largest computer company knocked on its door, and made a similar – but far shrewder decision. IBM was in a hurry to take delivery of a new OS so Microsoft bought one instead, and then tuned it up for IBM's use. The name of the program that Microsoft purchased was DOS, and it licensed tens of millions of units of DOS to IBM and the clone vendors over the many years that followed, providing the foundation for the then-tiny company's phenomenal growth.

software for use on computers running Microsoft OSs, as well as to upgrade their products to run on new versions of DOS and Windows as Microsoft released them.⁹ Until the rise of the clones, ISVs needed to create a new version of their software for each of the new small computers that were rapidly being offered in the marketplace, since each used its own, proprietary operating system. With the rapid domination of DOS-based PCs and PC clones over Apple and other brands, ISVs could develop a single version of a product that would run on a far larger market of business, and then home, computer users.

Up until this point in time, software development, and particularly software development among individual developers, had been a largely collaborative process. In these early days of mass access to computers, self-described “hackers” viewed each other as fellow travelers on a common adventure, discovering and furthering a brave new world of IT-based innovation in a collaborative process. Computer enthusiast clubs and cultures therefore took root, particularly around universities like MIT, where these hobbyists often had access to the source code of commercial software, and could readily trade software with each other on floppy disks for exploration and experimentation.



*Microsoft Staff, 1978
Gates, lower left; Allen, l. rt.*

All that now began to change.¹⁰ As might be expected, ISVs had no incentive to make the source code for their products available to anyone. They also hired lawyers to draft licenses that would prohibit users (and competitors) from reverse engineering their products. As venture capitalists invested in these new companies, non-disclosure agreements followed, as did non-competition agreements (except in California, where they were made illegal). Soon, a cone of secrecy, and often litigation, settled over commercial software development. Not everyone was pleased.

At the same time, a second OS was becoming pervasive not only on Intel x86 computers, but on another new class of computers: the very successful “minicomputers” and work stations developed and sold by a host of companies such as Digital Equipment Corporation, Apollo, Sun Microsystems and Data General. The origins of that OS trace back to 1969, when several Bell Labs employees decided to continue work, largely on their own initiative, on a scaled down version of the

⁹ Microsoft would only go so far, however, in enabling the efforts of other application software vendors, guaranteeing that Microsoft would retain advantages over its competitors (e.g., Microsoft could always release new versions of its application software before its competitors could whenever Microsoft released an OS upgrade). These, and other tactics that took advantage of Microsoft’s ownership of the OS upon which PCs, and later servers, ran resulted in long battles with regulators in the United States, and later Europe. For a readable version of the very competitive development of the PC software marketplace, see my (as yet) unfinished eBook, [ODF v. OOXML: War of the Words](http://www.consortiuminfo.org/standardsblog/index.php?topic=20071125145019553), at <http://www.consortiuminfo.org/standardsblog/index.php?topic=20071125145019553>

¹⁰ Most famously, a young Bill Gates sent an “[Open Letter to Hobbyists](#)” in 1975, objecting to casual copying of Microsoft BASIC. While Gates’ unhappiness was justified, the tone of his letter, which included the sentence, “As the majority of hobbyists must be aware, most of you steal your software,” invoked a significant reaction.

software ([Multics](#)) that was the subject of a collaborative project from which Bell Labs had just withdrawn. Over time, their work increased in scope.

At the time, AT&T (and therefore Bell Labs) was subject to a regulatory consent decree dating back to 1949 that prohibited AT&T from, “engaging...in any business other than the furnishing of common carrier communications services.” Because AT&T could not directly commercialize OS software beyond its own internal use, there were few in-place impediments to the developers sharing it with others. When AT&T’s legal department eventually became aware of what was happening, they consulted the language of the consent decree – and opted to straddle the regulatory fence. The lawyers concluded that UNIX (for that was the name the developers had given their work) could still be made publicly available, but only on request from academic and research institutions. Over time, word – and copies of the rapidly maturing OS – spread. By mid 1975, UNIX was officially in the hands of 33 labs, universities and secondary schools around the world, and doubtless on other computers on an unofficial basis.

Happily for AT&T, the trademark of the new OS did not have to be made freely available, and this provided the owner of that mark (which changed over time) with the legal right to limit the ability of anyone to use the UNIX name in connection with their product unless it conformed to the specification associated with the UNIX name (and unless they paid a trademark licensing fee as well).

Over time, UNIX became the OS foundation of choice on minicomputers and workstations (and later servers) the world over. Each vendor customized the original OS, but each vendor that wished to make use of the UNIX name could only go so far. The result was that UNIX-licensed OSs retained enough similarity – and therefore compatibility – with other UNIX licensed OSs to make it easier for a customer to migrate among them than it could among the mainframe computers sold by IBM and other vendors.¹¹ As a result, customers were less locked in to a single UNIX vendor, and ISVs were attracted to the growing market for UNIX-compatible application software.

There were other forces at play as well that ran counter to the increasingly proprietary nature of the marketplace. With so many PC and mini computers running the same, or a similar, operating system, there were incentives for ISVs in particular, and some hardware vendors, to facilitate the use of software across the hardware of multiple vendors without the need for expensive adaptation to run on each one. And with the advent of intense price competition among PC clone vendors, there were clear benefits to make as many parts (e.g., disk drives, connectors, memory chips, and so on) interchangeable as possible. The result was an increasing desire to develop open standards that would allow both hardware as well as software from different sources to be acquired and assembled into single, interoperable systems.¹²

The result was the evolution of a partly open (via open standards) and partly closed (through patents and licenses) marketplace in which vendors could control their

¹¹ Usually, these proprietary versions were given names ending in “X” to indicate their lineage (e.g., AIX, developed by IBM, and HP-UX, from Hewlett-Packard).

¹² The process continues today, turbocharged by the Internet-enabled desire to potentially connect every computer, everywhere, to every other computer.

products, while customers could enjoy a degree of competitive bidding between vendors. But developers could only collaborate among their co-workers, and customers were still subject to a large degree of lock in, especially in the case of software.

The Stallman revolution. Revolutions, by definition, are launched by revolutionaries, and often ones that have little patience for those that do not share their passion and their vision. The FLOSS revolution has been no exception to this rule, and certainly it has not lacked for proponents infused with the passion of their convictions. Some twenty-odd years after the genesis of the FLOSS movement, that emotional appeal continues to motivate tens of thousands of FLOSS advocates, even as OSS – its non-ideological stepchild – gathers increasing momentum in the (at best) values-neutral world of the commercial marketplace.

The universally acknowledged ideological founder and standard bearer of the FLOSS movement is [Richard M. Stallman](#) (more often referred to among the faithful simply as rms), a *magna cum laude* graduate of Harvard College who became exposed to computer programming while still in high school. In the years that followed his undergraduate years, Stallman pursued graduate studies, and then worked, in the hacker-based culture of the Artificial Intelligence (AI) Lab at MIT. But the culture of the AI Lab soon began to change: employees were asked to sign non-disclosure agreements, and several of Stallman's co-workers left to pursue higher-paying jobs in two high-profile startup companies that were formed to focus on AI technology: [Lisp Machines, Inc.](#) and [Symbolics, Inc.](#) Stallman became increasingly militant in opposing these restrictive practices, and disappointed that fellow-hackers would forsake the open sharing of software for stock options in for-profit companies.¹³



Richard Stallman, 2009

In the fall of 1983, rms formed what he called the "GNU Project," the name being a recursive acronym of the statement "Gnu's Not Unix!" The concept behind the project was to develop a new OS that would be similar to, and mostly compatible with, UNIX, but which would include no proprietary UNIX code. And it would be available in source code form as well to anyone that wished to study it, share it, modify it, or publish their own modifications. In 1984, Stallman quit working at MIT to dedicate his full time to the GNU Project, unencumbered by the intellectual property rights (IPR) rules applicable to MIT employees. In 1985, he founded the Free Software Foundation as a non-profit, charitable organization to provide a vehicle to hire programmers to work on the GNU Project.

¹³ Richard Stallman is as well known for the rough edges of his personality as he is for the brilliance of his thought leadership. Famously, he will not provide an interview to a journalist that does not agree in advance to use several of Stallman's preferred FLOSS terms. For example, the journalist must use the phrase "GNU/Linux" to refer to the operating system more commonly referred to simply as "Linux," in order to acknowledge that the GNU Project came first, and that non-embedded "Linux" implementations will invariably include software developed by the GNU Project as well.

Stallman's project attracted both individual as well as academic and corporate contributors, and completed a variety of tools, components and libraries. It also incorporated other free software components into its evolving OS where available, such as the X Window System, a graphical user interface developed by the [X Window Consortium](#).¹⁴ But while the GNU Project eventually completed or incorporated most components of a complete OS, for a variety of reasons, some technical, it failed to complete a fully functional and reliable "kernel" for its operating system (i.e., the software that serves the most basic core functions of an operating system).

At the same time that Stallman was managing the GNU Project, he was also maturing and codifying his ideology of software development and use. To Stallman, software was not simply a technical tool for managing or gaining value from computers, but a creative work that should stand on a par with literary and political works. Those that created computer code should also enjoy the same freedoms that journalists and other authors enjoy, unconstrained by rules of secrecy that would be intolerable if the creative work in question was a political essay, as compared to a compiler or text editor. While Stallman had no objection to an author (or other owner of the copyright in computer code) making money from computer code, he believed that the ideas and creativity captured in that code should be part of a greater commons.

As Stallman's thinking matured, he also came to believe that there should be limits on anyone "free riding" on the efforts of what commonly came to be called the work of the "community." If someone wished to make use of the labors of others by modifying code contributed to the community, they should be willing to make their own modifications available for the common good as well, at least if they hoped to make money reselling the modified work, as compared to simply making internal use of it.

To Stallman, software was not simply a technical tool for managing or gaining value from computers, but a creative work that should stand on a par with literary and political works

Stallman summarized his thinking in the early days of the GNU Project in the [GNU Manifesto](#), first published in Dr. Dobbs' Journal in March of 1985.¹⁵ In that manifesto, Stallman described his motivation as follows, under the heading, "Why I Must Write GNU:"

I consider that the golden rule requires that if I like a program I must share it with other people who like it. Software sellers want to

¹⁴ The X Window System was an ambitious software development project hosted by MIT with industry support. The author assisted in the spin out of the Project in 1993 as an independent non-profit membership consortium, represented it until its eventual merger into The Open Group in [1989], and assisted in the development of one of the first open source licenses, now known as the "MIT License."

¹⁵ The [GNU Manifesto](#) can be found in its original form, but with clarifying annotations, at the GNU Project Web site at: <http://www.gnu.org/gnu/manifesto.html> Additional material describing the Free Software philosophy can be found at [this page](#) of the Free Software Foundation Web site: <http://www.gnu.org/gnu/manifesto.html>

divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way. I cannot in good conscience sign a nondisclosure agreement or a software license agreement....So that I can continue to use computers without dishonor, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free....

Stallman expanded on this theme in a later section, titled "Why Many Programmers Want to Help:"

Many programmers are unhappy about the commercialization of system software. It may enable them to make more money, but it requires them to feel in conflict with other programmers in general rather than feel as comrades. The fundamental act of friendship among programmers is the sharing of programs; marketing arrangements now typically used essentially forbid programmers to treat others as friends. The purchaser of software must choose between friendship and obeying the law. Naturally, many decide that friendship is more important. But those who believe in law often do not feel at ease with either choice. They become cynical and think that programming is just a way of making money.

By working on and using GNU rather than proprietary programs, we can be hospitable to everyone and obey the law. In addition, GNU serves as an example to inspire and a banner to rally others to join us in sharing. This can give us a feeling of harmony which is impossible if we use software that is not free....

Stallman spent a great deal of time in his manifesto responding, in Q and A format, to potential criticisms of his ideas from an economic perspective. He dispensed with the interests of existing OS vendors in a way that would, understandably, hardly endear himself to them:

GNU will remove operating system software from the realm of competition. You will not be able to get an edge in this area, but neither will your competitors be able to get an edge over you. You and they will compete in other areas, while benefiting mutually in this one. If your business is selling an operating system, you will not like GNU, but that's tough on you. If your business is something else, GNU can save you from being pushed into the expensive business of selling operating systems.

Indeed, in this regard, Stallman's words proved to be prophetic. Microsoft CEO Steven Ballmer later compared the appeal of FOSS to that of communism, but one after another of the major non-OS software developers (eventually) came to



GNU Project Logo

embrace the economic advantages of utilizing the open source development model, at least selectively.

Stallman took greater pains to justify his new development model and morality to individual software developers, providing the same answer in many ways, in response to the same question ("How will I make a living?") also asked from various perspectives. Still, he offered no concessions to individual programmers, either. Here is one example:

Q: "Won't programmers starve?"

A: I could answer that nobody is forced to be a programmer. Most of us cannot manage to get any money for standing on the street and making faces. But we are not, as a result, condemned to spend our lives standing on the street making faces, and starving. We do something else.

But that is the wrong answer because it accepts the questioner's implicit assumption: that without ownership of software, programmers cannot possibly be paid a cent. Supposedly it is all or nothing.

The real reason programmers will not starve is that it will still be possible for them to get paid for programming; just not paid as much as now.

And in a later response:

It is not considered an injustice that sales clerks make the salaries that they now do. If programmers made the same, that would not be an injustice either. (In practice they would still make considerably more than that.)

*Free software is a matter of
liberty, not price*
- Richard Stallman

But perhaps Stallman's most radical statement appears at the end of yet another formulation of the same question ("Won't everyone stop programming without a monetary incentive?"). That response ends as follows:

What the facts show is that people will program for reasons other than riches; but if given a chance to make a lot of money as well, they will come to expect and demand it. Low-paying organizations do poorly in competition with high-paying ones, but they do not have to do badly if the high-paying ones are banned.

Over time, Stallman provided further articulations of his beliefs, including the Free Software Definition that he first published in basic form in the GNU Project Bulletin in February of 1988. In its current, more definitive, form, it reads [as follows](#), (explanatory text not included):

Free software is a matter of liberty, not price. To understand the concept, you should think of free as in free speech, not as in free beer.

Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it means that the program's users have the four essential freedoms:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

Stallman's second major contribution was on the legal front. Rms realized that in order for his ideological thoughts to be enforceable, they would need to be supported by legal agreements. He consulted various legal experts, but in the end personally drafted the licenses that he believed instantiated his vision, using plain English. The first was the GNU Public License ("GPL"), which was usefully updated two years thereafter.¹⁶



Current GPL version logo

Over time, the GPL v.2 became a very popular and widely implemented license, and the software (including Linux) that was released under it came to have great commercial and strategic value. As a result, when Stallman decided that it was time to revise the GPL again, the process was very long and convoluted, involving stakeholders of many types (both free software advocates as well as lawyers representing multinational IT corporations), with kindred spirit Eben Moglen, a Columbia Law School Professor and a founder of the Software Freedom Law Center, by turns channeling and serving as an intermediary between Stallman, who had the final word on all changes, and those that were on the four revision committees proposing, arguing for, and ultimately together recommending final changes. The new version of the GPL (v.3) was eventually released on June 20, 1987.¹⁷

¹⁶ The [original version](http://www.gnu.org/licenses/old-licenses/gpl-1.0.txt) of the GPL, adopted February 1989, can be found here: <http://www.gnu.org/licenses/old-licenses/gpl-1.0.txt> The [GPL v.2](http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt) is here: <http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt> The text of the [GPL v3, and related links](http://www.gnu.org/licenses/old-licenses/gpl-3.0.txt) and information, is maintained by the Free Software Foundation on this page: <http://www.fsf.org/licensing/licenses/gpl.html>

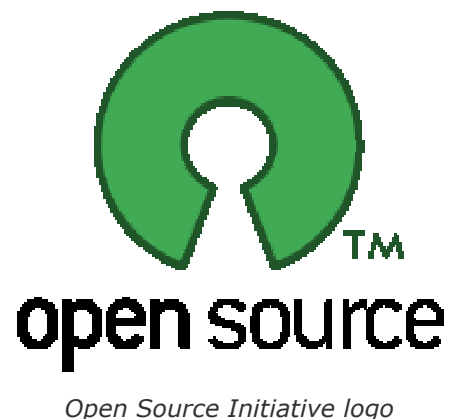
¹⁷ Like most other members of the FOSS cast, Moglen is an outsize personality. He is also a very skilled public speaker, and has become a major spokesperson for the free software movement. His

Unlike most legal documents, the GPL in any of its forms is as much a community social covenant as it is a binding legal agreement. It sets forth the norms of conduct that it expects from those that will use the code that is made available under it, and also imposes good behavior requirements on its licensees. As a result, it is a source of some puzzlement for lawyers conditioned to seeing documents of a very different kind accompanying software. But it is also a kind of Declaration of Independence and Bill of Rights that continues to have an expanding impact.

Today, almost 25 years after Richard Stallman issued his GNU Manifesto, proprietary software is still legal, and programmers are making more money than ever creating it. But at the same time, more FOSS software than ever is being written, and often, as predicted by Stallman, by programmers working for free, on their own time, in exchange for the satisfaction and status they derive from identifying with, and participating in a community of like-minded, committed professionals.

The OSS alternative: Richard Stallman dedicated great time and effort to articulating and spreading his message, gaining many supporters and becoming the most visible leader of what came to be called the “free software movement.” But while many resonated with some or all of his message, not all agreed that the development and use of proprietary software was immoral. In time, those who were attracted to the collaborative model of open source development and the availability of source code, but not to the full free software ideology or Stallman’s often confrontational style, decided to part company with the free software movement. Thus it was that an ideology that claimed the right to “fork” a software program as a basic freedom encountered a political schism of its own.

The initial spokespersons of this new branch of software development thought were, most visibly, Eric S. Raymond and Bruce Perens. Like Richard Stallman, they gave a label to their preferred mode of software development, calling it “open source,” thereby focusing on the factual availability of code, rather than on its ethical implications. And, like Stallman, they also founded (together with Jon “maddog” Hall and Larry Augustin, among others) a non-profit organization that incorporated the term that they had coined. Now programmers could rally to the call of either of two organizations: FSF, or the new Open Source Initiative.



Raymond and Perens share several attributes with Stallman: each is a skilled and prolific writer, well able to articulate his thoughts in ways that help gather adherents to their views.¹⁸ But their ideologies are different (in contrast to Stallman’s willingness to impose limitations on personal conduct for community

[personal Web page](http://emoglen.law.columbia.edu/) at Columbia University Law School can be found here: <http://emoglen.law.columbia.edu/>

¹⁸ Like Stallman, Raymond also has some less than mainstream beliefs. His Wikipedia entry notes that he is “a neopagan [and] an avowed anarcho-capitalist.”

benefit, Raymond is a Libertarian). Each became uncomfortable with Stallman's emphasis on a code of software morality, while approving of the capability of the collaborative process to develop better code faster. When Netscape, locked in what proved to be mortal combat with Microsoft, announced in February, 1998 that it would make the source code of its browser software freely available, Raymond and his co-founders announced the formation of OSI to recognize and promote open source development as a process in which proprietary as well as free software developers alike could participate and find value.

Unlike Stallman, who would presumably not mind the disappearance of proprietary software from the face of the earth, the OSI founders took what they believed was a more pragmatic approach that was inclusive of traditional business interests. Indeed, they welcomed those interests to the party. Not surprisingly, this has led to a degree of periodic friction between the open source and free software communities. Officially, however, the Free Software Foundation positions the definitions and goals of the two movements as being complementary rather than antagonistic.

Raymond became the principal spokesperson for the open source concept, but Perens made a crucial contribution as well, acting as the principal draftsman for what the OSI founders called the "open source definition." While similar in some respects to Stallman's Free Software Definition, the definition the OSI adopted is more lengthy and detailed, in part because it seeks to establish the parameters within which a variety of different licenses can exist that can be created and proposed by anyone.¹⁹ In contrast, the Free Software Definition maintained at the GNU Project Site is at any time supplemented in detail by only a few licenses, each developed and copyrighted by FSF.

The development, maintenance and (most importantly) application of the Open Source Definition to specific license agreements proved to be OSI's great contribution. By defining the business boundaries of open source rather than strictly regimenting its definition, and by appointing itself as the arbiter of which licenses fit within that definition, the OSI achieved several important effects.

Unlike Stallman, the OSI founders took what they believed was a more pragmatic approach that was inclusive of traditional business interests

First, it allowed a fairly broad spectrum of permissions and obligations to fit within its definition, allowing a wider range of business models to be accommodated. As importantly, it allowed squeamish corporate attorneys to find their comfort zone by granting them the ability to submit the licenses they created (sometimes with only legally inconsequential or idiosyncratic differences from already approved licenses) to OSI for approval. The result, as discussed in the next section of this article, was a proliferation of OSI-approved licenses, many of which are seldom used in practice – but also a more rapid spread of FOSS software as well, including programs

¹⁹ The [Open Source definition](http://opensource.org/docs/osd) appears as an Appendix at the end of this article, and can be found on line at: <http://opensource.org/docs/osd>

released, or contributed to, by traditionally conservative, multinational IT companies.

The rise of the Linux distribution: While rms and the GNU Project were making progress on many fronts with their free, alternative operating system, progress on one key component was lagging: the kernel that Stallman had named “Hurd.” That component has to date still not been released in robust form, in part because in 1991 a student in Finland who had just purchased a new Intel 386-based computer decided to hack a kernel himself, for his own amusement and education. Famously, he posted a note to an Internet user group page on August 25, 1991 that began as follows:

Hello everybody out there using minix —

I’m doing a (free) operating system (just a hobby, won’t be big and professional like gnu) for 386(486) AT clones.... I’d like to know what features most people would want. Any suggestions are welcome, but I won’t promise I’ll implement them :-).

At the time, virtually no one paid attention to the note. But the student who posted the modest announcement proved to be not only an extremely talented programmer, but a master in an entirely new discipline: managing the type of globally distributed, highly individualistic volunteer work force that the spread of Internet availability was only then making possible. His name was Linus Torvalds, and the name he gave to his little program, in honor of the UNIX program with which it was intended to be largely compatible, was “Linux.” Soon, people were flocking to the project, and their work was made available almost immediately to anyone who was interested in watching what was going on, using what was being created, or joining in themselves.



Linus Torvalds

The success Torvalds demonstrated in managing his increasingly ambitious project revealed the viability of an entirely new way to develop software – virtually and over great distances, allowing a critical mass of kindred spirits to gather to perform projects that never would have been launched if they were dependent on a local talent pool that needed to work under the same roof. Eric Raymond, of OSI fame, later described and popularized the new model in a seminal work he published in 1997, called *The Cathedral and the Bazaar*, which contrasted the traditional, slow-moving and methodical software development model (the “Cathedral”) with the more chaotic, real time, “rough consensus and running code” development approach taking hold on the Internet (the “Bazaar”).²⁰

²⁰ *The Cathedral and the Bazaar* has appeared in print in a number of media, including in book form. Eric Raymond maintains a [Web page](http://www.catb.org/~esr/writings/cathedral-bazaar/) with links to the full text of the essay in multiple languages, as well as additional information, at: <http://www.catb.org/~esr/writings/cathedral-bazaar/>

To the amazement of traditionalists, the Bazaar was capable of producing very good code – and quickly. Over the years that followed Torvalds' simple announcement, thousands of programmers from around the world contributed millions of lines of code to the project. The fantastic popularity of the Linux kernel effectively eliminated the need for the GNU Project to complete Hurd in order to achieve its ultimate goal of enabling a complete free and open OS.

With the emergence of such a well-supported OS becoming evident and the demonstrated success of the distributed FOSS development model, a new type of project began to proliferate. These projects were formed to assemble not just an OS, but a complete set of commonly desired application tools (e.g., word processors, spreadsheets, and so on) from the growing number of project-generated tools available, all well matched, and all licensed on FOSS terms. The result was that customers could download an increasingly wide range of free, useful, “out of the box” computer software environments from any Internet connection.

These “distributions” (“distros,” for short) were often tailored to a particular type of user requirement, from ambitious (e.g., for high-powered business users) to narrow (e.g., to serve as an “embedded” system, perhaps buried under the hood of a car, invisible and unknown to its owner). The Linux kernel could invariably be found at the core of these distributions, together with some or most of the GNU Project's components as well. Stallman's Free Software Foundation asked users to call them “GNU/Linux” distributions, but predictably, the marketplace preferred the shorter “Linux,” and in truth the GNU Project had also incorporated significant pieces of software developed by others (e.g., the X Window interface) without adding their names to the “GNU” handle.

Many of these new distribution projects were also formed on a non-profit basis. But entrepreneurs were figuring out how to make money from open source as well, most frequently by offering distribution and support services that their customers were willing to pay for – just as rms had supposed they would in his 1985 GNU Manifesto. When five-year old Linux distribution vendor Red Hat launched its wildly successful IPO in 1998, investors noticed.

What was perhaps more interesting was the way the realities of business and FOSS passion accommodated each other. The mixing of what might have seemed like oil and water was accomplished through the application of roughly comparable amounts of economic support, career opportunity and respect.

One model that emerged was a partnership of a semi-autonomous development project (sometimes legally separate, and sometimes not) that developed a FOSS product that could be downloaded for free, and a sheltering, supporting for-profit business that offered additional support and services (and sometimes additional software as well) to commercial customers for a fee. When this model flourished, the development project would have many participants that were not employees of



Logo of Red Hat, the most commercially successful Linux distribution

the for-profit business at all. The Linux distributions supported in this fashion that are currently most commercially successful are Fedora, supported by Red Hat, and SUSE, supported by Novell, for enterprise users, and Ubuntu, supported by Canonical, a Linux distribution optimized for easy use on laptops and other personal computers. But there are very well supported distros that are entirely the product of non-profit, community efforts as well, including the Gentoo and Debian distributions.

Linux could be assembled with FOSS programs in other common and useful ways as well, with the best known being its inclusion in a “solution stack” deployed on the great majority (i.e., millions) of Web application servers in use today. That stack is commonly referred to by the acronym “LAMP,” taken from the initials of its principal components: Linux, Apache HTTP Server software, the MySQL database package, and any of the following: PHP, Python, Perl (each, a Web scripting language).

Microsoft and FOSS: Richard Stallman had made no bones over the fact that an OS vendor would not be likely to be a big supporter of his GNU Project, and there is no company on earth that is more revenue-dependent on OS sales than Microsoft. Moreover, open source software could encroach on other Microsoft products as well, including the second major leg of its revenue stool, Office, the dominant productivity software package in use today.

Still, some (including Microsoft) did not take FLOSS seriously for many years, in part because they doubted that people would really produce complex software for free, and in part because the most visible advocate for FLOSS was Richard Stallman, who on occasion would arrive at public speaking engagements wearing a Biblical robe and, on his head, an ancient computer disk, to provide a halo effect reminiscent of a Renaissance painting.

This lack of early credibility perhaps was crucial to the success of FLOSS, since it allowed the FLOSS culture to grow largely unmolested in the dark, left alone by competitive forces, well-meaning but misdirected lawyers seeking to constrain rights in the software being created, or journalists unable to properly understand or describe what was in fact emerging.

This lack of early credibility perhaps was crucial to the success of FLOSS, since it allowed the FLOSS culture to grow largely unmolested in the dark

By the time the commercial marketplace began to be aware that FOSS was proliferating everywhere – including in their own datacenters – the reality of FOSS development was already well proven, widely practiced, and broadly implemented.

Moreover, some major IT vendors began to grasp the strategic advantages that they could reap from even wider use of FOSS. Most notably, IBM announced in December of 2000 that it would invest \$1 billion in enabling its products to run on Linux, and would assign 1500 of its engineers to create Linux products and services. The reality of so large and traditional an industry player as IBM making such a substantial strategic and economic commitment to FOSS was enough to boost the credibility of FOSS in general, and of Linux in particular, overnight.

But IBM sold hardware as well as software, and was moving forward with a business plan directed at greatly increasing its revenues from services as compared to products. The move also made sense because the largest number of dollars that IBM customers spent on any vendor other than IBM frequently went to Microsoft. A dollar in an IT budget not spent on Windows was therefore a dollar that was once more up for grabs. Other vendors, such as Sun Microsystems, Hewlett-Packard, Hitachi and NEC decided that Linux could mesh well with their business models as well.

With almost all of its revenues deriving from software sales and services, of course, Microsoft was in a far more challenging position. Once it realized that FOSS could represent a true threat to its businesses, Microsoft spoke out publicly against FOSS, questioning its quality, total cost of ownership, security and stability. Over time, however, customers found that the most popular FOSS products consistently were of very high quality, were usually cheaper to own, could be readily modified, were more secure, and were constantly being updated and improved. Moreover, they enjoyed more price competition among competing vendors, and could change vendors far more easily.

Microsoft's efforts to undermine Linux had a predictable effect in the FOSS community. The dominant software vendor was already unpopular in some developer quarters, and during the course of a long-running government antitrust investigation many stories of hardball commercial behavior that had long been rumored were confirmed and reported publicly. Microsoft thus made itself into a public enemy against which the FLOSS and OSS communities could unite. The many Web sites that focused on FOSS became,

and remain, perpetually awash in stories that report, or speculate, on what Microsoft might be up to next.



The present: Today, FOSS is firmly entrenched in many infrastructural niches, and gaining ground in applications as well. In the office productivity space, a variety of desktop (e.g., OpenOffice) and cloud (such as Google Docs) applications are gaining ground. Hundreds of millions of blogs run on WordPress software, and the vast majority of Web servers run on the LAMP stack. Meanwhile, SourceForge, the most popular free host for FOSS projects, reports that over 230,000 FOSS projects (not all active) reside on its servers. In the public sector, governments in many parts of the world (e.g., the European Union) look increasingly favorably on FOSS in their procurement activities, and more and more of the smartphones and other mobile, wireless devices that will outsell lap and desktop computers in the years to come run on one of a variety of flavors of Linux.

Most tellingly, even Microsoft appears to be turning a corner, with more and more of its internal teams turning to, or at least being open, to FOSS, if only because their customers give them no choice.

III Open Source Licenses and Legalities

As with each other major topic discussed above, the legal theories, agreements, and documentation that relate to FOSS are far too complex to explore more than superficially in an article of this type. But for current purposes, it is less important to acquire a deep knowledge of FOSS legal terms than it is to gain insight into why the legalities of FOSS are so important.

Threshold facts and principles: The interaction of FOSS and the law will seem like a hopeless jumble unless a few important facts are first understood:

- ✓ **No one entity usually owns a FOSS product:** For reasons that are partly historical and partly ideological, in most projects every individual developer that contributes a line of code will retain ownership of that line. As a result, any piece of FOSS is likely to have many authors. Since successful FOSS software undergoes constant improvements and corrections, the authors and their contributions are constantly changing as well. When you license a piece of FOSS, you therefore license not from one licensor, but from many – and perhaps from thousands.
- ✓ **It takes a village to build a distro:** FOSS comes in many pieces, large and small: text editors, libraries, interfaces, and much more. Because the store of available FOSS becomes ever larger and there is no reason to constantly reinvent the wheel, a FOSS application or OS is likely to include many modules and components borrowed from other projects as well as software developed by the project whose name is primarily associated with the application, OS or other software. (The same, incidentally, is also true of many proprietary packages, on which more below.)
- ✓ **Rock, paper, scissors:** Because of the heterogeneous nature of such “village” software, it will be likely that selections made on purely technical merit will result in a final package that includes modules that were developed under different license agreements. As a result, care must be taken to ensure that all of the licenses that relate to the constituent parts of the final package are at least minimally compatible, which can result in something of a dilemma at the end of a development project unless good legal hygiene was practiced along the way. (The problem is more severe for proprietary software vendors, because the developers that they employ are just as likely to be attracted by available FOSS tools and components as their FOSS project brethren – and may even participate in FOSS projects themselves. Such FOSS “contamination” can be particularly problematic if a FOSS component is included that was developed under a license containing a provision intended to trump all more restrictive terms applicable to any part of any product into which it is inserted.²¹)

²¹ As can be imagined, the availability of an ever expanding cornucopia of FOSS components has resulted not only in headaches for vendors, but in nightmares for anyone involved in a technology merger or acquisition. Any transaction involving (especially) a software vendor now includes a meticulous, line by line, “codebase” analysis of all software that the acquirer will purchase, to determine what license terms apply to that line. Where issues are (almost inevitably) discovered, remediation then follows, which may include removal or replacement of code that is available only

- ✓ ***It's not just about money:*** The "restrictive" licenses discussed below include provisions that are intended to protect the strongly held personal beliefs of the developers that prefer such licenses. As a result, these licenses include rules that are social as well as economic in origin and effect.
- ✓ ***New wine in new bottles:*** While some parts of FOSS licenses are quite similar to their proprietary analogues, and are included for similar purposes (e.g., warranty disclaimer language), other parts seek to accomplish new and novel results, often using language that is unfamiliar to lawyers. As a result, such licenses present new interpretive questions to judges, not all of which have, as yet, been answered. Where they have been addressed, it has been in the isolated courts of various jurisdictions. As a result, "settled law" may not be achieved on some new types of FOSS licenses for some time.
- ✓ ***It takes more than a license:***

Because any piece of FOSS software is likely to have many authors, each retaining ownership of her code, it is important for "contribution" (or similarly titled) documents to be in place between each developer and the project in which they participate, to be sure that a licensee of the software that the project makes available actually has the rights that they need. The process of collecting such rights is complicated by the fact that many contributors to a product may be subject to agreements with their employers that may be inconsistent with the rights needed by the project.

As can be seen, launching and maintaining a successful FOSS project, and distributing and using the results, takes place on a rather complicated legal landscape
- ✓ ***It's often all about copyrights:*** While some licenses (such as the GPL) include terms that are relevant, or that seek to influence behavior relating, to patents, the contribution agreements and distribution licenses that relate to FOSS often deal primarily with copyright and trademark rights. In contrast, a software license between two commercial entities may also give the licensee certain rights under patents that the licensor owns or controls. This dichotomy is in part due to the fact that those that contribute code to FOSS projects are not likely to control all patent claims that may be infringed by the use of the code that they contribute (and their employers may not, either). As a result, while the user of a popular FOSS product may have meaningful protection against an infringement suit as a practical matter, due to the market taboo against asserting patents against such programs, and the great number of allies that will spring to the defense of anyone that is attacked, a FOSS licensee does not receive the kind of warranty and indemnification protection that it might receive under a license to proprietary software. There are exceptions (e.g., in the case of some Linux distributions), in which a commercial distributor of the program chooses to

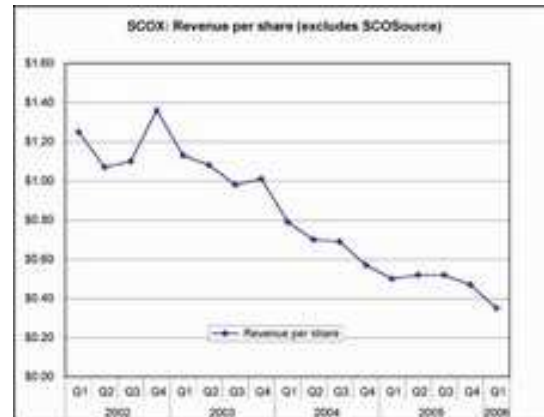
under terms deemed not to be acceptable by the acquirer. Well-run software vendors have adopted detailed internal procedures intended to document, and as necessary, avoid the inclusion of software subject to such terms, and new companies (such as Black Duck Software) that provide vetting products and services have emerged to serve the needs of developers and M&A firms alike.

step in and provide the same kind of warranty and indemnification protections to its customers.

As can be seen, launching and maintaining a successful FOSS project, and distributing and using the results, takes place on a rather complicated legal landscape. Unfortunately, some of the thousands of FOSS projects that have been launched (particularly in the early days) have paid comparatively little attention to the legal niceties that can make later use of their work product less problematic. However, best practices and supporting legal documentation for FOSS development are now broadly available, including at common FOSS project hosting sites, and such instances are therefore becoming less frequent.

SCO and the advent of FOSS Discipline:

In an example of “good news/bad news,” the FOSS development community became aware of the importance of maintaining legal discipline before the number of important FOSS projects began to proliferate greatly. The source of this raised awareness was a legal assault by a company called SCO Group. In June of 2002, SCO filed the first in a series of legal actions against implementers of Linux, alleging that the program contained “line by line” copies of SCO’s proprietary UNIX software code.



SCO's stock price. It kept going down

What followed was a (still ongoing) series of high profile and vigorously waged battles involving SCO, on the one hand, and Novell and IBM on the other, in two separate suits.

The background for the SCO litigation is quite factually complex, in part as a result of the multiple transfers (some poorly documented) of ownership that occurred over many years involving the copyrighted code of UNIX, and certain underlying patent rights. In part due to its tactics, in part due to the target (Linux – the poster program of the FOSS community), in part due to allegations that Microsoft was secretly funding SCO’s campaign, and in part due to the great antipathy that many members of the FOSS community hold for software patents in general, SCO was, and continues to be, excoriated as the Great Satan of the FOSS world.²²

The good news, however, was that it was plain for all to see that paying attention to such annoyingly bureaucratic practices as collecting signed contribution agreements and checking the header files of contributed software files, no matter how small, really was important. Despite the fact that the courts have consistently found that SCO’s allegations were groundless, the shock therapy applied by its

²² SCO management’s seeming strategy of “litigation as a business” has not paid off well for the company’s stockholders. SCO was twice delisted by NASDAQ, and ultimately filed for bankruptcy, after spending millions of dollars per fiscal quarter on legal bills over the past seven years. As of this writing, the bankruptcy trustee is deciding whether to permit the company to reorganize, or to force it into liquidation.

boorish and anti-community behavior provided a lesson that may have averted many disasters of real consequence that otherwise might have followed.²³

Permissive and restrictive licenses: The first licenses identifiable as “open source” licenses were very simple indeed, and focused on what a licensee could do, rather than what she couldn’t. The common ancestor of all open source licenses is generally agreed to be the “BSD License,” used to distribute a “UNIX-like” OS created by the Computer Systems Research Group at the Berkeley campus of the University of California. That OS was initially based on UNIX code, but over time, the code was gradually replaced, resulting in a non-identical, but functionally similar version of UNIX (in other words, because it was new code, it would resemble, but not infringe the copyright of the UNIX OS).



Logo of BSD Linux

The simple text of the original BSD license read as follows:

Copyright (c) 1988, Regents of the University of California
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * All advertising materials mentioning features or use of this software must display the following acknowledgement:
This product includes software developed by the University of California, Berkeley and its contributors.
- * Neither the name of the University of California, Berkeley nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS AND CONTRIBUTORS BE LIABLE FOR ANY DIRECT,

²³ The SCO suits had a variety of ripple effects throughout the FOSS world. One was the shelving of a very promising joint venture, called United Linux LLC, that had been founded to accelerate the global adoption of Linux by allying the vendors of four of the primary Linux distributions. The intention was to create a common distribution that could be supported on a global basis by each of the four participants: Caldera, [SuSE Linux](#), [Turbolinux](#) and [Conectiva](#). Large vendors, such as IBM, could then distribute the software to their customers with greater assurance. Unfortunately, the CEO of Caldera was later replaced by a new CEO named Darl McBride, who had a very different vision of how to make money out of Linux. Not long after, Caldera changed its name to the SCO Group – and all further activity at one of my most promising FOSS clients ground to a halt, never to be revived.

INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

As can readily be seen, the license says little more than, “Do as you please, but let people know where this code came from – and don’t sue us.” The BSD License, and the multiple, similar variations that followed, therefore came to be called “permissive” licenses.²⁴

The availability of increasingly valuable code under permissive licenses had two predictable consequences: people used the code, and some of them used that code to make money. Since the license did not require commercial users to make their new code available on a similar basis, however, two more predictable consequences followed: First, some licensees made the code they wished to sell available under proprietary licenses, and second, many developers that might have otherwise been happy to work gratis on code made available under a permissive license instead went to work for the high-paying, commercial vendors. Soon, there were multiple commercial, proprietary descendents of BSD Unix, including the version that helped launch Sun Microsystems.

Among those who were troubled by this behavior was Richard Stallman, and as we have already seen, he vowed to stop it. What Stallman did was to craft a license that would impose conditions on the reuse of FOSS that would map to the community values that he espoused. The result was a new type of license – the GNU General Public License, or GPL,

The first open source licenses said little more than “Do as you please, but let people know where this code came from – and don’t sue us.”

already mentioned in the previous section of this article. The GPL was intended to impose obligations on the user that would guarantee that any modifications created by the licensee would be available on terms that would comply with the Free Software Definition (quoted earlier), principally by requiring that the licensee would distribute source code as well as binary code, and by prohibiting the licensee from adding any restrictive conditions that would subvert developer freedoms.

Stallman gave the name “copyleft” to the provisions that accomplished the more innovative of these goals. Because a license including copyleft terms restricted licensees to release their modified works only under license conditions that were in harmony with the terms of the original license, this category of agreements came to be called “restricted” licenses.

²⁴ Later versions of the BSD license eliminate the third bulleted term. The most commonly used permissive license today is usually thought to be the simpler “MIT License” drafted by the author and Bob Scheifler, the Executive Director of the X Consortium, which developed the user interface adopted by the GNU Project and still used in Linux distributions today. The text of [that license](http://opensource.org/licenses/mit-license.php) can be found here: <http://opensource.org/licenses/mit-license.php>

The evolution of the GPL: The first version of the GPL was based on precursor licenses used with GNU Project component programs, and allowed multiple FOSS programs to be combined more easily. It was first used in 1989, and amended in 1991, at which time a somewhat more permissive license adapted for use with libraries was also released. That license was initially called the Library General Public License, or LGPL (which began life as version 2, not 1, in order to match the new version of the GPL, with which it harmonized), and later as the Limited General Public License.

The major evolutionary change from the first to the second version of the GPL was the addition of a broader, more free-form legal obligation that Stallman called the “Liberty or Death” clause, a kind of catchall hammer clause intended to thwart any action that might result in a program downstream from the original, GPL licensor ever being able to restrict the required freedoms of its own licensees.

The third (and current) version of the GPL was not released until March of 2007, following a long and laborious process of discussions, public comments, and multiple discussion drafts. The reasons for the protracted process were several, reflecting the increasing commercial value of FLOSS programs released under the GPL (most notably, Linus Torvalds had switched the Linux kernel to GPLv2 from its earlier, permissive license), a desire to thwart certain types of behavior deemed by Stallman and others to be anti-community that had occurred in the marketplace that the terms of GPLv2 had been unable to prevent, the large number of changes and goals that the revision was meant to incorporate, and the systemic challenge of having so many cooks, both lawyers and non-lawyers alike – over 130 in all, broken up into four committees – operating within the textual confines of the same, very small kitchen.

The result was a license that was applauded by many, but not all, constituencies. As a result, while new FLOSS projects are more likely to be launched under GPLv3 than v2, and an increasing number of pre-existing projects are, or already have, migrated from GPLv2 to v3, some projects have either preferred not to migrate, or have been unable to do so, due to logistical difficulties (e.g., the inability to track down no longer active programmers owning the copyright to their included contributions).²⁵

The GPL is unique in a variety of ways, not least among which is its iconic status as both a legal document as well as a social contract among those that adopt it. The result is that any legal challenge to the GPL would be deemed to be an attack upon the entire global community of FLOSS developers. Because so many commercial interests, and their customers, now rely so heavily upon FLOSS that is subject to the GPL, the license enjoys something of a legally untouchable status, and has attracted far fewer legal challenges than would normally be expected for such an innovative and broadly utilized license.

²⁵ As of this writing, the Linux kernel remains under the GPL v2, in part because Linus Torvalds does not agree with some of the new terms added to GPL v3, and in part due to the fact that even a desired migration would be problematic, in light of the fact that the whereabouts of so many previously active, but no longer contributing, owners of kernel code are not known. Consequently, their permission to change the terms under which their code could be licensed cannot be easily obtained.

Other licenses: There are many other FOSS licenses in existence today, largely as a result of a process of evolution not unlike what can be observed in the biological world. When FOSS first appeared on the scene, new project sponsors sometimes tinkered with the license of a predecessor, leading to the development of an evolutionary license tree that grew out of the original BSD terms of use. With the advent of the GPL, the same process was repeated, with proprietary vendors being particularly disposed to crafting their own variations on the GPL theme. Typically, the vendor or project that produced a new license would submit it to OSI for approval and entry on OSI's master list of approved licenses (other definitions and lists are maintained by, for example, the Debian Foundation).

Over time, many of the branches on these initial trees withered, resulting in a number of early licenses effectively being abandoned (i.e., they are rarely selected by new projects, and existing projects that originally made use of them may have migrated to a different license). Today, it is commonly acknowledged that there are more OSI-approved licenses (64 active and 10 licenses that have been superseded or retired are currently listed at the OSI site), with too few differences, than makes objective sense. In fact, only a handful of licenses are employed by the vast majority of all projects. They include the following: New and Simplified BSD, MIT, Apache 2.0, Mozilla 1.1, Common Development and Distribution License, Eclipse Public License, GPLv2 and V3, and the LGPLv2 and v3.



Apache

Logo of the Apache
Foundation

There have been a variety of efforts to create taxonomies of FOSS licenses in order to make it easier for new projects to select the license that best meets their goals, and for existing ones to understand the consequences of accepting code subject to other licenses.²⁶

Open source practice: While the goals of FOSS are straightforward, the legalities are not. Merging code together from multiple sources into both FOSS as well as proprietary products creates many traps for the unwary, both on the business (e.g., code contamination) as well as the legal (misunderstanding what a given license term is intended to address) side of the house.

The GPL (versions 2 and 3) in particular seek to achieve a variety of significant, and sometimes subtle, legal and commercial goals, and the elements of the social contract underlying these terms is as important as the strict legal terms. Concluding that a particular type of behavior is "legal" while ignoring the way in which the behavior may be regarded in the FLOSS community can result in undesired community, if not necessarily legal, consequences.

Finally, as noted, the body of case law that is available for courts to consult in crafting their own decisions relating to these new licenses is, as yet, scant. For all these reasons, anyone doing business in the open source area, and their legal

²⁶ OSI's [category page](http://opensource.org/licenses/category) can be found here: <http://opensource.org/licenses/category>

counsel, should take the time to understand the emerging lore, as well as the law, before they become too-deeply involved.

IV The FOSS Ecosystem

As the production and utilization of FOSS has expanded, an increasingly varied ecosystem has evolved to promote the use of FOSS, and to support and protect those that develop and use it. The principal components of that ecosystem are as follows:

Projects and Foundations: At the top of the pyramid are the projects and foundations that develop and maintain the most respected and implemented FOSS packages, such as Web server software (the Apache Foundation), browsers (the Mozilla Foundation), the Linux kernel, the most popular Linux distributions (e.g., Fedora, SuSE, Debian, Ubuntu, Gentoo, Mandriva, and so on), and other widely adopted protocols and software (e.g., Samba and OpenOffice). Other foundations have been formed to support broader goals, such as the Eclipse Foundation, which provides a “commercially friendly” OSS development environment.



As earlier noted, projects may be supported by independent, non-profit corporations formed for that purpose (e.g., Mozilla and the GNU Project), by for-profit companies (e.g., the Fedora project, by Red Hat; Ubuntu, by Canonical; and the OpenOffice project, by Sun) that have revenue-producing businesses based upon the FOSS projects they support, or by no one at all, as is the case with the vast majority of all FOSS projects, most of which are hosted by source code repositories.

Source Code Repositories: Developing and distributing FOSS requires almost no supporting infrastructure at all, as was demonstrated by the rapid success of the Linux kernel project, which was almost accidentally launched by Linus Torvalds. Given the plummeting cost of server space, the proven ability of highly distributed development teams to form and work together, and the viral way in which IT information spreads on the Web, the only real challenge to launching a new project is attracting and holding the interest of its volunteer participants.

The conjunction of community spirit (and sometimes Web advertising opportunities or indirect corporate benefits) has therefore led to the launching of a variety of free hosting platforms, usually referred to as “source code repositories,” that provide not only server space and visibility to volunteers, but a variety of other services as well, such as development tools, legal contribution agreement forms, the opportunity to designate a distribution license, and a communication platform. Perhaps the best known of these repositories is SourceForge, which was launched in late 1999, and as of early 2009, reported over 2,000,000 registered users and more than 230,000 active (and inactive) FOSS projects. Other hosts, both generic as well as more likely to host projects in certain IT areas, include RubyForge, LaunchPad and JavaForge.

Commercial Code Repositories: Not surprisingly, conservative, traditional corporations (and their legal staffs) have found the wild and woolly world of SourceForge projects to be disconcerting, at the same time as they have often found the output of such efforts to be commercially appealing. The result has been the creation of foundations for what might be thought of as the hosting of “domesticated” OSS (although not FLOSS) projects, with the foundation providing rule sets, legal agreements, code review, road map parameters, and other infrastructural items that give for-profit sponsors comfort that the output of the development projects that they underwrite or launch will meet their perceived needs, while still seeking to provide enough independence that non-employees will also wish to contribute. Two bear special mention:

Eclipse Foundation: EF was launched in late 2001 by IBM with the support of multiple partners, including Red Hat and SuSE. Its purpose was to host the creation of open source software development tools, based upon a platform that IBM had developed and made available as open source for that purpose. As stated in EF’s first press release, EF would provide:



Eclipse Foundation logo

...a common set of services and [establish] the framework, infrastructure and interactive workbench used by project developers to build application software and related elements...The Eclipse Platform provides source code building blocks, plug-in frameworks and running examples that facilitate application tools development. A complete sample plug-in based integrated development environment for creating Java applications (JDT) is included. Code access and use is controlled through the Common Public License allows individuals to create derivative works with worldwide re-distribution rights that are royalty free.²⁷

However, EF was not initially formed as an independent organization. While it was managed by a “Board of Stewards,” IBM retained various rights that somewhat limited the appeal of the project to the marketplace at large. Eventually, IBM agreed to spin EF out as an independent legal entity. The new, self-governing organization was announced on February 2, 2004, with a membership and governance structure reminiscent of a standards consortium, but with additional features adapted to accommodate open source technical participation on a more independent and community accessible basis.²⁸ EF became much more successful after this transition. Today, it boasts many active projects in a variety of areas, and its membership has doubled to more than 160 companies (both large and small), non-profits and universities.

²⁷ [Eclipse Foundation Forms to Deliver New Era Application Development Tools](http://www.eclipse.org/org/pr.html), Press Release (November 21, 2001), at: <http://www.eclipse.org/org/pr.html>

²⁸ [Eclipse Forms Independent Organization](http://www.eclipse.org/org/press-release/feb2004foundationpr.html), Press Release (February 4, 2004), at: <http://www.eclipse.org/org/press-release/feb2004foundationpr.html>

CodePlex and CodePlex Foundation: In May of 2006, Microsoft launched a SourceForge-like site called CodePlex.com to host open source code development generally, and in particular to spur OSS development around its .Net framework.²⁹ That site has been successful, and today hosts many projects. As the latest in its tentative steps towards engagement with OSS development, Microsoft announced the formation of the CodePlex Foundation in September of 2009 as a new non-profit foundation.³⁰ The mission, governance structure, and level of independence of the Foundation are still evolving during a 100 day transition under a Microsoft designated interim Board of Directors.

The press release announcing the formation of the CodePlex Foundation states that it was formed:

...with the mission of enabling the exchange of code and understanding among software companies and open source communities...[and] as a forum in which open source communities and the software development community can come together with the shared goal of increasing participation in open source community projects. The CodePlex Foundation will complement existing open source foundations and organizations, providing a forum in which best practices and shared understanding can be established by a broad group of participants, both software companies and open source communities.³¹



Logo of Microsoft's CodePlex Foundation

As of this writing, the CodePlex Foundation is a work in progress, but one which demonstrates the continuing evolution of the marketplace as the traditional world of proprietary software comes to grips with the increasing importance of FOSS to IT customers.³²

FOSS vendors: While the FOSS development model has not (as yet) often provided the golden goose that venture capital investors briefly hoped it would, there have been solid investment wins in a number of cases, including Red Hat, the most successful Linux distribution vendor and still an independent, public company, and MySQL, a popular relational database management system owned and sponsored by a Swedish company, MySQL AB. MySQL was acquired by Sun Microsystems for approximately \$1 billion in February 2008.

For profit corporations do, however, reap great commercial value from FOSS, and in consequence provide an enormous amount of support for FOSS, not only through

²⁹ [Microsoft Announces CodePlex, a New Collaborative Development Portal](http://www.microsoft.com/presspass/press/2006/jun06/06-27CodePlexPR.mspx), Press Release (June 27, 2006), at: <http://www.microsoft.com/presspass/press/2006/jun06/06-27CodePlexPR.mspx>

³⁰ CodePlex Foundation home page, September 10, 2009.

³¹ Ibid.

³² I have written two extensive analyses of the initial governance structure and business plan of the CodePlex Foundation, which appear in this issue of *Standards Today*. They originally appeared in the Standards Blog, where they can be found here:

<http://www.consortiuminfo.org/standardsblog/article.php?story=20090914102959510> and here: <http://www.consortiuminfo.org/standardsblog/article.php?story=20090930123746629>

direct monetary contributions to projects and supporting institutions (such as the Linux Foundation), but by allowing (or directing) their employees to participate in FOSS projects, which in turn redounds to their own benefit in a variety of ways (e.g., by gaining first-hand familiarity with code evolution as it happens and future direction as it is decided).

Major multinational companies that have made particularly visible contributions and strategic commitments to FOSS include Google, Hewlett-Packard, Hitachi, IBM, Oracle, Motorola, NEC, and Sun Microsystems, among many others.

Promotion, support and protection: An extremely varied mix of non-profit entities has been founded to support, promote, and protect either FOSS in general, or specific FOSS software products in particular. As a gross generalization, they fall into two categories: those founded by groups of individual developers, sometimes with corporate support, and those founded by corporations, usually hoping for community participation.



Logo of MySQL ABS

Two organizations of the former type that have already been discussed in detail are the **Free Software Foundation** (FSF) (formed to promote the Free Software Definition and support the GNU Project, and later undertaking additional work, such as the evolution of the GPL and other FOSS licenses), and the Open Source Initiative (OSI) (formed in part as a delayed reaction to FSF, and principally known today as the repository of the Open Source Definition and the list of OSI-approved OS licenses).

Several organizations of the corporate-initiated kind have been formed to either protect, or to demonstrate the determination to protect, FOSS from perceived enemies. The formation of such well-funded entities was important in the early days of the commercialization of Linux, due to the potential impact of the SCO suits on prospective customers (SCO originally sued four Linux end users, although those cases were stayed while SCO, Novell and IBM engaged in serial legal warfare). These organizations continue to play important, though different, roles today for two reasons.

The first is an area of rising concern, represented by the proliferation of patent "trolls" (i.e., non-vendor companies that develop and patent technology – or simply purchase patents – for the purpose of demanding the payment of patent royalties from those that do produce products or provide services that would be infringed by the patents). The patent licensing business model is currently on the rise, and is likely to continue to cause angst to the FOSS community as new mechanisms (e.g., patent auctions) are introduced to help the owners of inactive patents monetize these intangible assets.

A more variable concern arises from the periodic veiled, and sometimes not so veiled, threats by Microsoft to assert what it claims are 235 patents infringed by popular FOSS software, such as Linux, OpenOffice, and email messaging programs.

Examples of these defensive, and sometimes offensive, organizations are the [Linux Foundation](#), which was formed in 2005 as a result of the union of the **Free Standards Group** (which maintained the Linux Standards Base) and **Open Source Development Labs** (OSDL), formed in 2000 for a variety of purposes supportive of Linux adoption by enterprise users. Beginning in June of 2003, OSDL provided financial independence to Linus Torvalds, by making him an “OSDL Fellow,” allowing him to dedicate his full time to supporting the Linux kernel. OSDL also became the funding vehicle for a “Linux Defense Fund” when the SCO litigation erupted onto the scene. That fund (still in existence) was formed to underwrite the cost of providing legal counsel for Linus Torvalds and other kernel developers in the event that they were drawn into the SCO litigation.

Today, the Linux Foundation provides a broad variety of promotional, educational, supportive and protective functions, including reimbursing travel expenses for key kernel developers, organizing and hosting community meetings and conferences, developing and hosting a variety of Web sites (including Linux.com, which it acquired in early 2009), hosting development of discrete activities (e.g., the [FOSSBazaar](#) informational site and the [Moblin](#) mobile Linux operating system project), and much more. The Linux Foundation is funded primarily by membership fees paid by its corporate members in three categories of membership (Silver, Gold and Platinum) with ascending fees.



A for-profit entity formed exclusively for protective purposes is the [Open Invention Network](#) (OIN), established in 2005 by IBM, Novell, Phillips, Red Hat and Sony with a very substantial initial capitalization in order to fund the purchase of patents that might otherwise be asserted against Linux or a variety of other important, Linux-related FOSS software. Once purchased, these patents can then be asserted defensively by member FOSS users against any companies that might allege that FOSS programs they use infringe the third party’s patents. OIN seeks to increase the scope of its net of protection by recruiting additional members, and by entering into cross licenses with non-member companies under terms that allow all members and cross-licensees to use the growing pool of owned and licensed patents in the event that they are sued.

More recently, OIN has engaged in additional activities, sometimes on a collaborative basis with other Linux ecosystem members (e.g., the Linux Foundation), such as the “[Linux Defenders](#)” program, which seeks to weed out poor quality patents before they are allowed to issue, and “[Linux Defenders 911](#),” which provides a place for commercial and community members to seek assistance if they are “victimized” by parties “antagonistic to Linux and true innovation.”

At the opposite end of the corporate to Floss spectrum are organizations like the [Software Freedom Law Center](#), which has more of the character, and engages in activities more similar to, a legal aid clinic. The SFLC focuses to a much greater degree on the needs of the individual community developer and of FLOSS projects of any size, and provides free legal services to its non-profit clients. It also

provides helpful publications on FLOSS topics, such as FLOSS development best practices. In recent years, it has begun representing FLOSS projects in asserting the GPL against commercial company GPL licensees that may be violating its terms.

The Blogosphere: There are an astonishing number of sites, both traditionally commercial (e.g., [LinuxToday](#), which is owned by Jupiter Media), semi-commercial (e.g., [SlashDot](#), which takes advertising but very much has a community vibe at all levels) as well as non-commercial (e.g., [GrokLaw](#)) that focus significantly or almost exclusively on FOSS or FLOSS-related news. Some, like GrokLaw, have generated great respect for the thoroughness and consistency of their reporting over the years, and have become the “go to” sources of information on topics such as the SCO litigation. Many accept contributions of news (often in addition to publishing their own reporting), or pointers to news, allowing such sites to aggregate large amounts of information in real time for a global audience. As a result, not many facts (or, for that matter, rumors) of interest to the FOSS community escape the notice of Linux blogs and news sites. Such sites also often provide a rallying point for action plans to coalesce among FLOSS community members.

V Summary (and a Look into the Future)

The development of the FOSS phenomenon has been unique in a variety of ways, both social and legal, economic and political. While this process has been historically unique, what we have already witnessed may be but a harbinger of further revolutions of similar impact yet to come, as our traditionally industrial, nationally-based societies complete their transition into a more singular, globally

There can be little doubt that FOSS is here to stay, or that the commercial significance of software developed within the FOSS will continue to grow.

interconnected, technology based economy and society. If this is the case, then the careful study of the FOSS phenomenon as it has developed to date, and the close observation of how it evolves in the future, will be particularly instructive.

But in either case, there can be little doubt that FOSS is here to stay, or that the commercial significance of software developed within the FOSS process will continue to grow. What is less certain is exactly how the FOSS political, social and technical phenomenon will continue to evolve in the years to come. Some of the more interesting questions remaining to be answered include the following:

- ✓ **Political evolution:** Will the revolutionary zeal of Richard Stallman and his supporters sustain, or, as with so many social and political movements of the past, will it dissipate as younger developers, accustomed to the ready availability of source code, begin to take FLOSS for granted?
- ✓ **Developer leverage:** Today, the ability of software engineers from around the world to collaborate in the development of software of great commercial value has given them unprecedented power and independence over the direction of their efforts, relative to the multinational corporations that

increasingly rely upon FOSS projects to advance their own fortunes. This dependence by corporations on forces beyond their immediate control runs counter to their usual risk-avoidance goals and strategies. Will the unprecedented influence currently enjoyed by collaborative pools of individual software engineers ("labor") be maintained, or will the traditional advantages of corporations ("management") find a way to reassert themselves over time, resulting in the "capture" of the FOSS process? And if this happens, will it result in a "lose/lose" scenario, killing the creativity and energy that power the FOSS phenomenon?

- ✓ **Legal developments:** Some of the most commonly employed FOSS licenses (e.g., the GPL, versions 2 and 3) have innovative features that have as yet rarely been interpreted in court. Will they be upheld in court, and if not, how will the FOSS community revise these essential tools?
- ✓ **Government uptake and endorsement:** Many governments (e.g., in the European Union) are beginning to favor FOSS for social, economic and policy reasons. Will FOSS become increasingly preferred for government procurement, and if so, what impact will the exercise of such substantial buying power have on software development and commercialization practices generally?
- ✓ **Best practices:** While the best FOSS projects are extremely successful, they often depend on the skills and leadership of individuals. At the same time, the legal and economic status (e.g., unincorporated, incorporated, sponsored, and so on) of projects is as likely to have been the result of accident as design. Given the increasing importance of FOSS software and the likelihood that such programs will remain useful over long periods of time, the stability and success of FOSS projects will become increasingly important. Will best practices of formation, governance, management and distribution be compiled, and if compiled, broadly implemented?
- ✓ **Support:** Will questions such as these be answered as the result of haphazard evolution, or through thoughtful and respectful support from affected stakeholders, and if so, by which stakeholders?
- ✓ **Influence:** The principles underlying FOSS have broad applicability to, and have also borrowed from, other current movements involving "openness," in areas such as content (scientific information, music files, visual images, etc.) and government access and participation. How will these concurrent forces continue to affect, and be informed by, each other?
- ✓ **Patent law:** Software has been broadly patentable for the last decade in the United States, but not elsewhere. Today, there is great unhappiness among a surprising range of stakeholders over what are perceived to be (at best) flaws in the U.S. patent system. These range from the grant of overly broad patents to the costs and variability in outcome of lawsuits (often depending on which Federal district court a plaintiff chooses as the venue in which to bring its case). Some would go farther, and believe that there is no justification for applying patent protection to software inventions at all. Will the courts in the U.S. continue to narrow the gate through which successful

patents can be granted and successfully enforced, or perhaps overrule their own earlier holdings that software can be patented at all? If so, what will the impact be upon the software development industry, and how will it affect the future of FOSS?

The answers to questions such as these will be interesting indeed. If they are the right answers, they will have a substantial and positive impact on our economy as well. Needless to say, it will be in the best interests of all if individual developers, corporations, and governments around the world are aware of the mutual benefits to be gained from FOSS, and work together to nurture the unique and revolutionary process that has already produced such useful – and perhaps surprising – results.

Copyright 2009 Andrew Updegrove

Sign up for a [free subscription](#) to **Standards Today** at

At <http://www.consortiuminfo.org/subscribe/2.php?addentry=1>

Appendix:

Open Source Initiative Open Source Definition

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

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

3. Derived Works

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

4. Integrity of The Author's Source Code

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

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

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

8. License Must Not Be Specific to a Product

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

9. License Must Not Restrict Other Software

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

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

As accessed from the OSI site at <http://opensource.org/docs/osd> on October 12, 2009

Selective Bibliography:

Core Reading: While there are many academic and popular articles that relate to FOSS, the first-hand literature relating to the theory, origins, and significance of OSS, and especially of FLOSS, is more limited. The following list is of the latter variety, and includes a number of works that anyone should consider “must reads” to gain insight into FOSS from an insider’s point of view.

DiBona, Stone, and Cooper, *Open Sources 2.0: The Continuing Evolution* (O'Reilly Media 2005)

<http://www.amazon.com/Open-Sources-2-0-Continuing-Evolution/dp/0596008023>

Gay (ed.), *Free Software, Free Society: Selected Essays of Richard M. Stallman* (Free Software Foundation 2002)

<http://www.amazon.com/Free-Software-Society-Selected-Stallman/dp/1882114981>

Moody, Glyn, *Rebel Code* (Basic Books 2001)

http://www.amazon.com/Rebel-Code-Linux-Source-Revolution/dp/0738206709/ref=sr_1_1?ie=UTF8&s=books&qid=1257107286&sr=1-1

Raymond, *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (O'Reilly Media 2001)

<http://www.amazon.com/Cathedral-Bazaar-Musings-Accidental-Revolutionary/dp/0596001088>

Rosen, *Open Source Licensing: Software Freedom and Intellectual Property Law* (Prentice Hall 2004)

<http://www.amazon.com/Open-Source-Licensing-Software-Intellectual/dp/0131487876>

Salus, *The Daemon, the Gnu and the Penguin* (Reed Media 2008)

<http://www.amazon.com/Daemon-Gnu-Penguin-Peter-Salus/dp/097903423X>

Also consider:

Lessig, *Free Culture: The Nature and Future of Creativity* (Penguin 2005)

<http://www.amazon.com/Free-Culture-Nature-Future-Creativity/dp/0143034650>

Wheeler, David, *Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!* (Revised April 16, 2007)

http://www.dwheeler.com/oss_fs_why.html

Ongoing commentary: The number of Linux and FOSS-related blogs and news sites are legion. Here are three examples that provide active, ongoing commentary from a range of viewpoints:

Asay, Matt: *The Open Road* (CNET.news.com): *Matt is a prolific poster, and writes from the business/OSS point of view*

<http://news.cnet.com/openroad/>

Jones, Pamela: Groklaw: *PJ is a champion of the FLOSS viewpoint, and has been the "go to" authority for reporting on legal news of significance to FLOSS for years. Groklaw is particularly known for its thorough coverage of the SCO litigation. The site has a very active community of followers and participants, and PJ's posts always inspire extensive discussions)*

<http://www.groklaw.net>

Linux Foundation Blogs: *Includes the blog entries of multiple authors, including LF Executive Director Jim Zemlin, Community Manager Brian Proffitt, and (occasionally) Linus Torvalds. My blog entries are cross-posted to the LF site as well*

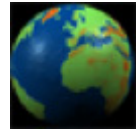
<http://www.linuxfoundation.org/news-media/blogs/browse-blogs>

STANDARDS BLOG:

Nurturing the Flower of Foss at the CodePlex Foundation

[Andrew Updegrove](#)

The following two blog entries were posted at the Standards Blog on September 14 and 30, following the announcement by Microsoft that it had launched a new, non-profit foundation to support the development of open source software. Due to Microsoft's historically antagonistic relationship with the free and open source software (FOSS) community, the announcement was greeted with skepticism by many within the FOSS community. In these two entries, I offered my views on the type of governance structures and business goals that the CodePlex Foundation would need to adopt in order to win the trust and participation of individual software developers.



The CodePlex Foundation: First Impressions (and Recommendations)

[Andrew Updegrove](#)



Well, it's been a busy week in Lake Wobegon, hasn't it? First, the Wall Street Journal broke the story that Microsoft had unwittingly sold 22 patents, not to the Allied Security Trust (which might have resold them to patent trolls), but to the Open Inventions Network. A few days later, perhaps sooner than planned, Microsoft announced the formation of a new non-profit organization, the CodePlex Foundation, with the mission of "enabling the exchange of code and understanding among software companies and open source communities."

Not surprisingly, more articles were written about the apparent snookering of Microsoft by AST and OIN than about the new Foundation. But while the tale of the 22 patents is now largely over, the CodePlex story is just beginning. Microsoft says that its goal for the new Foundation is to create an open and neutral environment, and that the formation documents posted and governance structure described at the CodePlex Foundation site can provide a foundation for such an organization. The CodePlex site also makes clear that the Bylaws you can find

there are just a starter set, stating, "Our governance documents are deliberately sparse, because we expect them to change."

That's good to hear, because I've reviewed all of the material at the CodePlex site, and I think that quite a bit of the governance structure will need to change before CodePlex can expect to attract broad participation.

Over the past 22 years, I've helped structure scores of open, consensus based consortia and foundations, and represented over 100 in all (disclosure: they include the Linux Foundation; a full list can be found [here](#)). In this blog entry, I'll show where I think the legal and governance structure of CodePlex has wandered off the open path, and offer specific recommendations for how the structure could be changed to give people (other than Microsoft business partners) confidence that CodePlex will be an organization worth joining.

Since there's a lot of ground to cover, to make it an easier read I'll use the self-interview approach that I've picked up from Steve O'Grady over at RedMonk.

Q: What's the sixty thousand foot guidance on how to set up an organization that will inspire confidence that it's safe to join?

A: It's all about three closely related factors: appearances, control mechanisms, and broad support. What you want to do is to create a structure that you demonstrably can't control. If you claim that you want the organization you launch to be neutral, and then people find "gotchas" in the documents, you've lost the credibility war on the first day of battle.

It also helps enormously to launch with multiple partners, rather than try to add them later after people are no longer paying attention. You'll never get more press than on the day you do your public launch, and if both competitors as well as allies are standing next to you on the stage as co-founders, that sends a powerful message that the organization really is not under any individual company's control.

For this reason, new organizations traditionally operate in stealth mode until they sign up an impressive roster of co-founders, so that people pay attention, and figure that there is broad industry support for what you want to accomplish. If instead you're out there all alone, then people wonder why that's so.

In this case, Microsoft launched without any co-sponsors (it has been theorized by many that the launch date was accelerated to offset the adverse publicity generated by the disclosure of the sale of the 22 patents), which I think was a mistake. If you go through the CodePlex site, you also learn that, while additional sponsors will be welcome, Microsoft has provided \$1 million in funding for the first year's operation. Microsoft will also provide the staff that will run the organization.

While it's good that Microsoft is willing to provide so much economic support in times like these, it's not helpful in building trust that the organization really will be independent and neutral. For better or worse, if all of the money and all of the staff come from one company, it will be hard for most folks to believe that CodePlex it will really be neutral in action.

Perhaps most significantly, when you go through the formation documents in greater detail, you also start running into “gotchas.” Some of these can be easily changed, and perhaps were meant to be open for discussion. But others (such as the decision not to form CodePlex as a membership organization) are so fundamental that I expect that Microsoft doesn’t intend for them to change.

The bottom line is that forming a successful consensus-based organizations is a bit like stepping through the looking glass – you win by giving things away, not by extracting value from others or controlling them. You have to create a place where people can be expected to conclude that it's safer to be a part of the organization, than to stay outside. Consequently, if it looks like you've kept too much control, the best you can hope for is to form a glorified user group. I’ve written extensively on how to form an organization that is convincingly open, for example [here](#) and [here](#).

The bottom line is that forming a successful consensus-based organizations is a bit like stepping through the looking glass – you win by giving things away

Q: So now let’s cover the basics; how is the Foundation set up?

A: Microsoft organized CodePlex under the non-profit laws of the State of Washington, which may be a good neutral choice, or may not. Most attorneys (myself included) aren’t familiar with Washington law, so it’s hard to tell (I always use Delaware law when forming a new non-profit, since its laws are very flexible, and most attorneys have some familiarity with it). Also, CodePlex has not been set up as a membership organization, which is very unusual for an organization operating in an area that usually relies on consensus in order to be credible.

Q: Is that good or bad?

A: In my view, it’s bad, because it means that the Board of Directors not only has complete control, but the Board is also self-perpetuating (i.e., the directors elect their own successors). Moreover, there are no term limits on how long a Board member can serve. In this kind of organization, the Board is not answerable to the participants, and the participants have no say or control at all over how the organization is managed or evolves.

Q: But as long as the Board is balanced, shouldn’t that be OK?

A: In theory, yes. And, to be fair, even in organizations (like most of the consortia I set up) where members elect the Board, almost all actions are approved by the Board, rather than the members. And in order to pay the bills, it's common that those that pay more get more of the board seats. But I always try to get the founders to agree to charge smaller companies significantly less than large companies to join at a membership level with board election rights, and also to allocate board seats to achieve diversity in whatever way is relevant to the particular organization (e.g., geographically, by industry sector, to include end users, and so on).

In this case, individuals and companies that decide to participate in CodePlex won't be able to vote for the directors at all. At minimum, this means that CodePlex will have to work very hard to convince others that the Board really is balanced, and therefore will look out for the best interests of all stakeholders, and not just the company that is paying all of the bills.

Q: Is there any way to tell from the documents how likely that will be?

A: There's one provision that particularly concerns me. Currently, the Board has six members, and the Bylaws provide that the successor board that will be appointed within 100 days will have only five members – that's a very small board indeed.

Q: Is that a problem?

A: In my view, very much so. Ordinarily, when you form an organization that will be consensus based and will need to satisfy many constituencies, you want to have a board that is large enough to provide lots of different voices at the table, without making it so large that it becomes unworkable. Well, if you start to think about balance, what would be the best you could achieve with a Board of five?

I wouldn't expect that the number of community representatives will outnumber the corporate representatives, so that means we could expect at least two corporate representatives besides Microsoft, and therefore no more than two community representatives. Out of the entire open source community, how would you allocate those two seats? One to a staff representative from, say, Mozilla, and one to an individual developer? If so, that means that there would be no room for a government representative, or an academic, or a standards community representative, or an end-user. Some of these groups are much more under represented in open source projects than software vendors, and also have a lot to gain from an organization like this – as well as a lot to share with the other members.

No matter how you slice it, you just can't get real representation of a market and user sector as diverse and broad as open source with a board this small.

Q: Can't the new board just change the number if it's too small?

A: Yes and no. Under the Bylaws, it takes a two-thirds vote of all directors serving in order to enlarge the Board. That means that it would take 4 out of 5 directors to add any extra seats. So if Microsoft directly or indirectly controls only two seats (e.g., if it holds one seat, and a sympathetic business partner holds another), they could block the Board from enlarging itself – ever. Moreover, the permanent board will be divided into three classes – one director will have a two-year term, two will have three-year terms, and two will have four-year terms (terms will be for three years thereafter). This is unusually long, and further insures that whatever Microsoft puts in place at the end of the 100-day planning period will have a very long life indeed. For all of these reasons, I'd say that in order for CodePlex to be credible, it will be essential for the permanent Board to be larger and more diverse than the initial Bylaws permit.

Q: The CodePlex Web site stresses that Microsoft will be taking recommendations from anyone and everyone. More specifically, it says:

Our Board of Directors is an interim board. While we've worked hard to see that the board has a community voice as well as a partner voice, we think one of the board's primary missions is to act as a search committee to find a permanent board of directors that brings representation from commercial software companies and open source communities such that all parties feel confident the board can fairly represent their views.

Q: Doesn't that help?

A: Yes, it certainly does, but we'll need to wait and see whom they pick. If you look at the list of interim Board members, the only people that aren't Microsoft employees are Miguel de Icaza (the business partner voice), and Shaun Bruce Walker, of DotNetNuke (the community voice).

Miguel is a very skilled open source manager, but Novell has had a rough ride these last few years, and its relationship with Microsoft has been very important to growing revenue and to its continuing survival. And Shaun's brief bio at the CodePlex site reads in part:

Shaun is the original creator of DotNetNuke®, an open source web application framework and web content management system for ASP.NET which has spawned the largest and most active open source developer community native to the Microsoft platform (700,000 members and 6.5 million downloads).

That doesn't mean that Shaun isn't a good, community kind of guy, but there certainly are many community representatives that Microsoft could have chosen that have no strong ties to the Microsoft ecosystem at all.

Given that gaining trust for openness has a lot to do with appearances, Microsoft would have been smart to have chosen people with no ties to Microsoft at all – especially given that the Microsoft representatives on the interim board, acting alone, will be able to vote in the permanent, self-perpetuating Board.

Q: But isn't there a Board of Advisors as well?

A: That's so (although I note that the Bylaws say that CodePlex "may" rather than "shall" have a BoA – not a good call from the appearances perspective). But note that the Bylaws say that the Board of Directors has no obligation to consult with the BoA or follow its advice. Moreover, the actual duties of the BoA can be set by the Board by vote – they don't need to appear in the Bylaws at all. So far, the CodePlex Web site doesn't display the minutes of any Board of Directors meetings, so we don't know whether the BoA may already have a charter (some, but far from all, consortia and open source foundations post Board minutes on their public Web sites. It would be a good sign if CodePlex decides to follow the more open practice of posting them).

The initial Board of Advisors is made up about 50 – 50 by Microsoft and non-Microsoft employees (you can find the full list in the right hand column on this page, but without links to their biographies so far). But again, that's only before you look at the past and current relationships of some of the non-Microsoft employees. Stephen Walli, for example, is a former employee of Microsoft, and has done consulting work for them since then. Stephe is a great guy, and a personal friend of mine. In fact, I'm confident that he will provide a strong and independent voice, so I think he's a great person to have on the BoA. But again, many other people don't know Stephe, so if Microsoft really wants to build confidence in its intentions, it should recruit people with no ties at all. Moreover, with a Board of Advisors, you want to bring in outside voices, not replicate those that are already represented on the Board of Directors. So there really isn't any reason to have more than one or two Microsoft employees on the BoA, if indeed any at all. If the permanent Board of Advisors still has multiple Microsoft representatives, that won't be a good sign.

Q: The CodePlex site says that it "is a 501(c)(6) organization." Does that help?

A: Not yet – and possibly seeking tax exemption wasn't even the original intention, since I note that the current Bylaws don't include the provisions that the IRS would expect to see there in a tax exempt organization.

CodePlex may now hope to become tax exempt in the United States under IRS Section 501(c)(6) (i.e., as a trade association, which is the right exemption for them to shoot for), but I can't believe that it has been granted that status yet. An applicant needs to fill out a detailed description of what it will do, how it will be structured, what its budget will be, and what its activities and charter will be. All of these details are to one extent or another yet to be determined. After CodePlex does nail down these details and send the application in, the IRS will take four to six months to give a first response (usually a list of requests for further information).

My firm and I have submitted applications for scores of organizations, and have sometimes had to go to great lengths to satisfy the IRS that an applicant meets the tax exemption criteria if it looks like a single member might be able to exercise too much influence. As CodePlex is currently structured, I think that Microsoft will have a tough time getting tax exempt status at all, because the IRS will not approve an organization that it believes has been formed to be too much for the unique benefit of one company, rather than for the benefit of the industry identified in the application.

In order to make that decision, the IRS looks not only to what the espoused mission of the organization is stated to be, but to what it can infer from the other details in the application. Typically, the IRS looks at who controls the Board, who provides the funding, and similar factors. This isn't to say that CodePlex won't be able to gain tax exemption, but I believe that it will need to make a lot of changes to what I see there now if it is now before it fills out its application and submits it.

Q: What about the CodePlex mission? How does that sound?

A: I had to smile a bit when I listened to the (scripted) interview at the site. The premise seems to be that (a) “some companies” have “culture” problems that keep them out of open source projects, or are “uneasy” with the “intellectual property” rules of open source foundations; (b) that “more companies” would participate “as much as they should” if better practices, and intellectual property tools, were developed; and (c) that a place is needed to bring “such companies” and open source developers together. It’s clear that all of these statements would be true if you substituted “Microsoft” for the phrase, “some companies,” but I haven’t noticed that any of these factors has been a problem for most other software vendors.

This slide from the interview will give you the flavor:

- Commercial software developers currently under-participate in open source projects
 - Cultural differences
 - Differing development methodologies
 - Differing perspectives on copyrights and patents
 - Differing perspectives on licensing
- No other foundation is dedicated to changing that situation

Some of these points would have been valid six years ago. But you’d have to look pretty hard today to find a major software vendor that isn’t significantly involved in open source activities. And almost all of whatever friction that may once have existed between employees of commercial vendors and solo developers mixing it up in open source projects has disappeared. Today, there’s a good, synergistic relationship, with employers realizing that it’s in their best interests to encourage their employees to participate in projects (and not try to closely control those employees when they do), and individual developers realizing that rising through the ranks of an open source project is a great way to get high-paying consulting gigs, as well as full time jobs that provide lots of freedom.

Q: What about the licensing tools posted at the site?

A: They’re not bad, if what you want to do is convey the right to create code that can be distributed under any flavor of open source license. But why would any developer or contributor want to sign such an all purpose license? Lines of code are contributed to defined projects, not to some code bank where they can be archived for posterity.

Developers want to know what the specific license is that their code will be subject to when someone uses it, and usually to have their name in the header of the contributed file as well. None of this is provided for in the templates that CodePlex is encouraging people to use, and the interview clearly states that promoting these licenses will be a goal of the organization. This line of text on the “Contributions, Licenses and Patents” slide set caught my eye as well:

Foundation will extend rights to all downstream developers and users of the contributed code

That's "downstream" as in "but not upstream," even though the next bullet reads that CodePlex will be "License-agnostic." Those two bullets are inherently contradictory.

And there's another problem here: the Bylaws again require a supermajority Board vote to change the text of these template contribution and assignment agreements from the way they read right now. If this was just a starter set of Bylaws, there would be no need for them to call out this particular type of decision for such a vote, so presumably this provision is also not meant to change.

Q: Why do these templates matter?

A: In two ways. First, the CodePlex site says that the Foundation will be promoting their use throughout the industry. Second, the site states that CodePlex is intended not only to develop and promulgate best practices, but to host open source projects as well. Unless CodePlex is set up in a truly neutral fashion, that will lead many people to worry that Microsoft wants to create and legitimize "their" kind of development environment, where Microsoft can feel safe launching projects (all of the initial projects under consideration are Microsoft projects) under IPR rules, and under licenses, that fit their view of what open source should be all about.

Whether it likes it or not, Microsoft is likely to be held to a higher standard with CodePlex than another company might, due to its historical hostility to open source, and to its current mixed messaging on the same topic. I expect that unless significant changes are made, many people will conclude that CodePlex is intended to become some sort of "alternative universe" of open source development, populated by Microsoft business partners, where only the more limited types of open source licenses are considered to be good options for developers to use. Those licenses are fine for some purposes, but most developers – and even commercial companies – don't choose them today. If CodePlex flourishes under this type of regime, I won't be surprised if Microsoft (as would most other vendors in the same situation) begins to tell customers that this type of patent-friendly environment is what open source software is "really" all about.

When you combine this with the assertions at the CodePlex site that a primary goal is to get more software vendor employees participating in open source projects across the board, you can easily see why the community might fear that CodePlex has been formed in part to recruit legions of new project participants that will have a new and different agenda than the existing members of the already existing projects that they join.

Q: So what's your bottom line?

A: There are a lot of games you can play when structuring an organization to make it look open, but still be sure that the founders will have a lot of control for a long time. It may simply be that Microsoft hired a firm to help it structure CodePlex that went overboard on trying to protect its client. But either way, the CodePlex documents have been set up in such a way that the Foundation's Board of Directors will have no accountability to anyone who participates in CodePlex. As they exist

today, they will also ensure that Microsoft will have ongoing veto power for many years to come even if they have only one board seat, and a friend in another.

In short, I think that the materials at the CodePlex site give a lot of cause for legitimate concern, both legally and from a public perception point of view. If Microsoft really wants CodePlex to attract more than its business partners as participants, I think that it needs to go back to the drawing board and make some drastic changes. Hopefully this is just a stumble, and they will be open to suggestions to do just that.

Q: What would you recommend?

A: If Microsoft really wants the open source community, as well as Microsoft's competitors, to believe that CodePlex is intended to be a safe, neutral place for them to spend their time and efforts, this would be my "must change" list:

1. Provide that the Board will have no fewer than eleven members.
2. Provide that no company and its affiliates (including Microsoft) can have more than one representative on the Board of Directors or Board of Advisors.
3. Provide for a distribution of Board seats by category in order to ensure a truly representative body. That distribution might mean only two seats for commercial software developers, two for open source foundation project managers (a Linux kernel developer lead would be a good choice for one), one for a large enterprise software user, one for a small to medium enterprise (SME), one for a government agency representative (this seat might need to be non-voting), and so on. By a simple majority vote, the Board would be able to change this distribution over time as the marketplace continues to evolve.
4. Establish appropriate membership classes with the right to nominate and elect directors.
5. Commit to an open membership policy, such that anyone can join, subject to meeting minimal, non-discriminatory eligibility criteria (I expect that this is already Microsoft's intention).

There are some additional changes that the Interim Board would be wise to consider implementing as well, each of which would greatly increase credibility and encourage participation:

1. Take back three quarters of the initial funding, and charge corporate members a fee to participate, in order to ensure that the organization is not dependent solely on Microsoft to pay the bills.
2. Provide for the formation of committees and working groups that will carry out the actual work of CodePlex within the strategic

plan established by the Board. These committees would develop and adopt deliverables that would be subject to final approval by the Board of Directors, but the Board's role would be limited to ensuring that proper processes have been followed, and that final deliverables are consistent with the initial charters of the working groups that created them.

3. Hire an outside management company to provide staff, rather than using Microsoft employees.

Yes, that's a lot of changes. But if there really is a need for individual developers and commercial vendors to get together in a new organization, then community members will need to feel like CodePlex is a safe place to be. Right now, I can't see that happening without some serious rethinking of the entire governance structure as currently proposed.

Further Reflections on the CodePlex Foundation:

The Glass Half Full

[Andrew Updegrove](#)



Two weeks ago, I wrote a critical analysis of the governance structure of the CodePlex Foundation, a new open source-focused foundation launched by Microsoft.

But what about the business premise for the Foundation itself? Let's say that Microsoft does restructure CodePlex in such a way as to create a trusted, safe place for work to be done to support the open source software development model. Is there really a need for such an organization, and if so, what needs could such an organization meet?

As with my last piece, I'll use the Q&A approach to make my points.

Q: Let's hear what Microsoft has to say before you start pontificating again. What is the mission of the CodePlex Foundation?

A: That's fair. As stated on the Foundation's "About" page, it is:

The mission of the CodePlex Foundation is to enable the exchange of code and understanding among software companies and open source communities.

Q: Hmm. Kind of general. Anything else to help us understand what Microsoft has in mind?

A: It looks like they expected that reaction, because the first Q&A exchange at the FAQ you can find at the site reads as follows:

Q: OK, I've read the Mission Statement. But what, exactly, is the CodePlex Foundation going to do?

We believe that commercial software companies and the developers that work for them under-participate in open source projects. Some of the reasons are cultural, some have to do with differing software development methodologies, and some have to do with differing views about copyrights and patents. In general, we are going to work to close these gaps. Specifically we aim to work with particular projects that can serve as best practice exemplars of how commercial software companies and open source communities can effectively collaborate.

The About page has additional detail, including the following:

The Codeplex Foundation also provides a channel of communication from the open source community back to Foundation partners and other commercial software companies, advancing the dialog between commercial software companies and open source communities.

Other goals include developing model contribution agreements for use by projects and providing a new venue to host projects. There are variety of other materials at the site you can review, including an interview with interim Executive Director Sam Ramji, a press conference file, and more.

Q: All well and good. But how will CodePlex be different from other open source foundations?

A: Now I get it. You've already been to the CodePlex site. Yes, they answered that question, too, like this:

Other foundations are targeted at particular projects, platforms, or applications, such as Firefox and the Mozilla Foundation, or Gnome and the Gnome Foundation. We wanted a foundation that addresses a full spectrum of software projects, and does so with the licensing and intellectual property needs of commercial software companies in mind. Having said that, we expect the Codeplex Foundation to be complimentary to, and not competitive with, other open source foundations. One measure of our success will be if other foundations experience an increase in participation from commercial software developers because of us.

Q: That second sentence sounds a little one-sided: “We wanted a foundation that addresses a full spectrum of software projects, and does so with the licensing and intellectual property needs of commercial software companies in mind.”

A: Yes, it does. It will be interesting to see whether the Foundation decides to stick with that emphasis or not. If it does, it might expect that open source community members will wonder why they should participate, and stay away (in droves).

Q: Any other breadcrumbs in that line of thinking worth noting?

A: Yes, there are others, but for brevity’s sake, I’ll note just this one, again, from the main FAQ:

Q: Why are you creating the CodePlex Foundation at this time?

We see a convergence of maturing technology and evolving business models - an inflection point - underway where more commercial companies are willing to participate in open source projects. We saw a great opportunity to drive change, and Microsoft was willing to fund a nonprofit foundation to make the change happen.

The open source community is famously unwilling to be “driven,” so again, I think the Foundation needs to do some rethinking here, and make it clear what’s in it for the community to participate if they really expect them to engage.

Q: OK, you’ve been uncharacteristically patient. How do you want to kick off the next section of this exchange?

A: Let’s talk about whether there really is a need for a new Foundation in the general neighborhood of the CodePlex Foundation’s stated mission, and if so, what that need might be.

Q: Fair dinkum, and it is your blog. So how about I ask you this: Is there a need for a new Foundation in the general neighborhood of the CodePlex Foundation’s stated mission, and if so, what might that need be?

A: Why thank you (cough <the check is in the mail>). Let’s start with a few observations about the current state of development of the open source model and the ethos of the open source movement. If you look at it from the historical perspective (which I’m always prone to do), you can compare the open source movement to any other political movement, which, to an extent, it is.

Q: A political movement? How’s that?

A: Well, it has a philosophy, it’s had dedicated leaders who have skillfully articulated that philosophy, and it’s inspired and rallied a broad and very large population of adherents to its banner. It’s also gone through a period of creative ferment, with passionate debates over both fundamental issues as well as finer points, developed schisms between camps with differing viewpoints, and soon.

Q: That sounds kind of unstable.

A: Yes, but necessary, especially in the beginning. As they say, democracy is messy. And despite the high level of dynamism, some of the largest and most conservative technology companies in the world (e.g., IBM) have realized how much value can be found in open source development and licensing, and have come on board. That's pretty remarkable, given how risk-averse huge companies generally are.

The really big question now is how the open source movement will evolve from here. Revolutionaries are necessary to bring about revolutions, but they're better with big ideas than with boring details. Problem is, it takes a fair amount of attention to boring details to turn grand revolutions into stable, lasting systems.

Q: OK, so how do you see that happening?

A: One possibility is that corporations will try to over-domesticate the open source development model and kill its spirit. If that happens, then you're likely to see less participation, and two separate types of projects: corporate projects where all or almost all of the participants are the employees of the group of companies that launch, and rely on, the project's output, and community projects, with largely independent developers participating, as well as corporate employees engaging for their own personal, avocational reasons.

Q: How do you think that would play out?

A: The corporate projects could work well as cost sharing mechanisms, allowing companies to develop software they want to see exist because it helps them sell other products and services. But the talent, creativity and passion would still migrate to the community projects, so corporations would never get the full advantage of the open source development model.

Q: So what's the other possibility?

A: Corporations continue to realize that the magic of open source comes from volunteerism, numbers of participants, merit-based advancement, openness, and other factors that can't be domesticated. You can't push a string, as they say.

Q: This is all very interesting from an academic perspective, I'm sure, but does this actually have anything to do with the CodePlex Foundation?

A: Indeed, it has everything to do with the Foundation. That's because (and to return to the political analogy), I think that the open source movement needs to mature, just as any revolutionary movement needs to mature, if it is to become a long term force.

A: What do you mean by "mature?"

Q: Passion is great to launch a movement, but it's often the boring details that have to be filled in to make it work long term. In the case of open source, the creative ferment that I referred to earlier has resulted in a somewhat chaotic

landscape – far too many licenses with far too few differences between them (the same is true with contribution and similar agreements), competing priorities between the interests of various stakeholders, and other fault lines that create legal issues that can slow down the spread of open source development and uptake.

If community members would like to see free and open source software rather than proprietary software predominate in the marketplace, then it's in their best interest to address these issues, and relax these tensions, as well.

Having a place where community members and vendors can get together and talk about these things could result in life being easier and better for everyone. By the way, I think that such a place needs to invite more than just individual and corporate developers, too, if it's really going to be useful. End users should have a seat at the table, as should foundation and project management, and also mergers and acquisitions experts – the current state of open source licensing at best adds many extra cycles to doing any kind of deal these days, and sometimes real nightmares.

A: So let's get more granular now. What kind of things do you think the members of such a foundation should engage on?

Q: Here are some issues that would be at the top of my list (asterisked items are already mentioned in some fashion at the CodePlex Foundation site):

1. There are too many licenses, too many of which don't mesh well. Let's narrow down the numbers and address where they don't fit well, and also document clearly and concisely where the problems are. There's already some good work going on in this area (e.g., at the Open Source Initiative), so in part this could be more of a supporting and promotional role.
2. Some of the existing, popular licenses that have been around for awhile could be cleaned up a bit, but whenever I help set up a new foundation, everyone is very reticent to touch the license they choose, because it's easier to reuse the version posted at OSI than to raise concerns over what has been changed, and why. So while recommending a subset of the existing licenses, let's suggest a few tune up changes to some of them as well.
- 3.* Involving corporate employees in projects can be a real pain, from the legal side. A project needs to be sure that it has the necessary intellectual property rights to any code an employee contributes, but that employee will often have already signed an agreement with her employer that says the employer owns the code and the related IP rights. In that case, you need the employer to sign something.

That's not too bad if the employer wants the employee to be participating, but what if the same employee works for, say, a multinational agribusiness? Who does the employee go to, and why should they say yes, particularly if they don't know anything about open source software? The concept of having well-recognized model contribution agreements that individuals and corporations would readily sign – as suggested by the CodePlex Foundation Web site – is not at all a bad idea, as it would facilitate getting such permissions. Of course, they

need to say something that's acceptable to the community as well as to the corporations.

- 4.* What about the rights in project code, as between a project and an individual code contributor? From a purely simplistic, legal perspective, the ideal situation is for contributors to transfer ownership of their contributions to the project, or to grant the project an unrestricted license to change, distribute, and sublicense the code (the latter route is the one taken at the current model Contribution Agreement to be found at the CodePlex site). Why ideal? Because this allows the project to exercise all rights needed to accomplish its goals, such as adopting a new version of its distribution license (the alternative is to go back and get permission from all of the individual contributors – many of whom may no longer be reachable). From the perspective of many developers, however, that's entirely unacceptable.

At one level, the question here is about the rights of authors (code contributors) versus the rights of publishers (projects), which is a conflict that's been around at least since the days of Gutenberg. But in this case – a distribution's authors are its publisher. Perhaps with time contributors may end up thinking more like publishers and this issue may go away, but for now, it would be great to make some practical progress on this one way or another – or in more than one way, so that people could make intelligent, project by project choices. (By the way, Stephen O'Grady's excellent Q&A on the CodePlex Foundation has a very good discussion on this topic).

5. Project governance is another area where developing best practices would be useful, making it easier for new projects to be launched with less effort, and in a form that will be most likely to facilitate success.
- 6.* I think it's always beneficial to help each community (corporate and developer) become more aware of the others' reality. There's no better way to appreciate another person's perspective and legitimate needs than to sit across a table (or listserv) from them and hash things out.
7. Needless to say, not every project is led by a Linus Torvalds, and even when one does come close, inevitably that person is going to want to retire or move on. Open source development projects are still in their relative infancy, with few important projects having gone full cycle from formation to some sort of happy ending (i.e., a transition to just a few, legacy users after the software has been superseded in the marketplace and the project has gone dormant). It would be great to review how the most successful projects have been run internally, analyze what made that happen, and then set up better "how to" manuals than currently exist. An element of this would be to consider how individual community members and dedicated corporate participants most productively interacted.
8. Last but not least, I think that having a place where the sometimes competing demands of open standards and open source could be discussed and worked out as well.

It would be great to see an organization that could tackle questions like these, either individually, or in some cases, in collaboration with existing organizations.

Q: What about hosting projects? The Web site talks about that as well.

A: I'm a bit ambivalent on that one. Do we need another project host? Probably not. But hosting projects as test beds to try out and demonstrate the type of deliverables that I mention above could be a very worthwhile endeavor. If those projects run really well, put out great software on an efficient basis, and attract and keep large, diverse communities of participants that work well together, that's a great way to develop, confirm, perfect and demonstrate the kind of best practices that others could benefit from.

Q: So I'm hearing from you that you don't think that the concept of the CodePlex Foundation is necessarily a bad idea, but that (a) it needs to be set up differently to inspire trust, (b) it needs to reset its priority statements to reflect an equal concern for community priorities as well as corporate needs, (c) it needs to change its messaging to demonstrate a commitment to provide convincing value for community members, and (d) while the initial list of deliverables is in the right ballpark, it could be further refined.

A: You know, I couldn't have said that better myself.

Bookmark the Standards Blog at <http://www.consortiuminfo.org/newsblog/> or set up an RSS feed at: <http://www.consortiuminfo.org/rss/>

Copyright 2009 Andrew Updegrove

Sign up for a free subscription to **Standards Today** at

<http://www.consortiuminfo.org/subscribe/2.php?addentry=1>

CONSIDER THIS:

#60 The Constantine Code and the *Missing Standard!*

Andrew Updegrave

One of the realities that every standards professional must deal with is the sad fact that everyone else in the world thinks that standards are...

[start over; no one else thinks about standards much at all]

Ahem. One of the things that standards folks must come to terms with is the fact that on the *rare occasions* when anyone else thinks about standards *at all*, likely as not it's to observe that standards are...

[how to say this as delicately as possible]

...boring.

[There. I've said it]

But really, now, this perception has got to change. And with the recent release of Dan Brown's latest pot boiler, *The Lost Symbol*, I believe I've figured out how to make standards really, really exciting. *Really.*

Yes, given the reading public's obsession with probing the backwaters (both real and imagined) of western history for amazing hidden truths, the way is clear for me to recount a tale of ancient intrigue and power to demonstrate that standards in fact lie at the core of life ~~the universe and everything~~ *[sorry; wrong author]*. Moreover, in the exciting, surprise ending, I'll also show that this amazing truth was *right there before your unseeing eyes all the time!* But unlike the italicizing Mr. Brown, I won't have to invent anything at all along the way.

The long-unrealized standards secret that I am about to share is this: even the most mundane dynamic and procedural feature of the modern, global process of developing, maintaining, branding and certifying standards has been in existence for almost 1700 years. Moreover, it can trace its lineage to the master plan of a powerful Emperor who invoked the standards process to protect the very existence of his empire from the threat posed by a rogue bishop and his clamoring followers bent upon igniting a standards war.

How's that you say? Or, as Dan Brown's Robert Langdon would of course phrase it, "*What the Hell?*"



Yes indeed, and it shouldn't take a symbologist to figure this one out. The year was 325 AD, the emperor was Constantine I, the rogue bishop was Arius, and this first global standards conference was the First Council of Nicaea. And as you'll see, the analogy of this convocation and its results to the modern standards process is as uncanny as any fictitious historical outtake dished out by Dan Brown. He just makes a *whole lot* more money when he does it.

So, at last, I invite you to consider this:

In the first few hundred years after the death of Jesus Christ, we are told, the gospel was spread far and wide throughout the western world by his apostles, and then in turn by their followers. Of course, this was a time when literacy was a rare commodity, so the words that spread the Word inevitably varied with the teller and the retelling. Moreover, Christianity was at that time suppressed in the Roman Empire, forcing many of the new faith's proponents, and their adherents, to spread the message verbally, and in hiding.

Only over time, therefore, was the story of Jesus' life and teachings written down. Not surprisingly, the facts contained in the various versions of his life that eventually were set down do not always agree. In fact, historians have concluded that none of what we know today as the Gospels were actually written by the apostles to whom they were attributed, and that no first hand account therefore exists.



Capitoline Museum bust of [Constantine I](#)

What we know today as the canonical books of the New Testament are therefore but a selection of the many accounts of the ministry of Jesus that were eventually recorded. Historians also tell us that the Gospels are not representative of the full range of accounts from which they were selected – an inconvenient truth that the emerging faith had no alternative but to address. Of greatest concern to the early Church of St. Peter was the fact that not all of the accounts that then existed agreed on some of the new religion's most foundational beliefs – including the nature of the divinity of Christ himself.

If this sounds surprisingly like parts of the story line of the *Da Vinci Code*, it should, because up until this point, Brown's tale tracks historical fact reasonably closely. But what Brown didn't mention on his way to best seller success is what actually happened next. Surprisingly, the actual events were in some ways consistent with his far more fancifully concocted story line.

And that was this: upon converting to Christianity, the Roman Emperor Constantine I took it upon himself to help define what "Christianity" actually meant – to come up with a single, empire-wide standard, if you will, that would codify the essential beliefs that defined the new religion, and replace the multiple and potentially divisive belief-specifications that had emerged, based upon the oral histories that had taken root over time. From among these several views of

Christianity, he decreed, would come a single belief-standard that, once universally implemented from East to West, would standardize all religious thinking and bind his empire more firmly together.

And so it was that Constantine convened the first global standards conference, inviting bishops from every corner of the empire to gather and apply their wisdom and inspiration to the divination of a single understanding of the Truth. And come they did, from Libya and from Gaul, from Persia and from Jerusalem, from every Roman province save Britain. Indeed, even from present day Georgia, across the Black Sea and beyond the boundaries of the empire itself.

In the best traditions of modern standard setting, the individuals who answered the call did not have to pay their own travel and lodging expenses (Constantine picked up the tab). Moreover, the all-expenses paid trip took the bishops to a pleasant watering hole at the lakeside city of Nicaea, Turkey, near the shores of the Bosphorus and sitting astride the exotic trade routes to the Far East. Notwithstanding the dangers and slow pace of travel in those far-distant times, between 250 and 318 bishops (accounts differ) answered the call out of the c. 1800 then serving, providing broad representation of all populations of stakeholders.



[Arius of Alexandria](#)

Nevertheless, representation was geographically lopsided, due to the location of the conference, and the reality that there were c. 20% more eastern than western bishops to begin with. The result was that every outcome of the Council was consistent with the positions proposed by the eastern bishops – providing perhaps the first example of regional block voting influencing the final outcome in a global standard setting process.

As is so often the case in standards development today, the starting point for discussions was the submission of proposals from various competing factions. On the subject of the divinity of Christ, for example, the leading candidates for adoption were presented, on the one hand, by St. Alexander of Alexandria and by Athanasius (*Jesus was the literal Son of God*), and by Arius (*Jesus was only the figurative Son of God*). Initially, sentiments were split, but, as with the ISO/IEC today, the process was run by consensus rather than majority vote. Near unanimity was eventually achieved behind the literalist submission, and all but two of those in attendance signed the final conference document supporting that position.

The conference attendees also reached other decisions, such as foregoing more straightforward, logical and easy to apply approaches to determine the date upon which Easter would fall each year, and settling instead upon the bizarrely convoluted mechanism still used today (the comparisons to modern standard setting are too painful to articulate).

And what of poor Arius? Just as today, the consequences of losing a standards war can be severe. Arius took his defeat hard, and refused to conform to the new

standard. And so he was excommunicated – denied certification as a Christian, because he failed to conform to the adopted standard.

The punishment of Arius thus also provides an early and instructive example of the power of branding in connection with standards. Since the Church owned the brand (“Christian”), once it incorporated compliance to the standard it had adopted to what were, in effect, the licensing terms that licensees (Christians and their priests) were required to obey, the church could deny certification to anyone that refused to comply. “Heresy,” after all, is simply a more judgmental term for non-conformance.

Still, Arius was lucky, compared to those that sought to peddle non-conformant beliefs in later days. Over time, compliance testing became much more rigorous, arguably reaching its high water mark during the Spanish Inquisition. In those days, the consequences of telling what even then was probably an ancient joke (“The nice thing about standards is that there are so many of them”) could be mortal. Happily, conformance testing techniques and penalties for standards non-compliance are today far more benign than the admittedly often effective, but no longer politically correct, tools of the past, such as the rack, the stake, and the *peine forte et dure*, or, as it was sometimes more prosaically and descriptively called, *pressing*.



Death by Pressing
Confess!

Despite the unavoidable reality that not everyone’s standard can be adopted, the First Council of Nicaea worked out just fine for most. The conference also turned out well for Nicaea, which apparently gained a reputation as a pleasant destination for standard setting junkets. Six more theological conferences were convened there, with the last being held in 787 – quite a nice run of business, to be sure.

Most impressively, the Catholic Church remained theologically unified around the standard adopted at this first great standards conference for almost two millennia (so far), notwithstanding the fall of Rome and its consequent division between territories east and west, and all of the turmoil in the world in the years that followed.

Indeed, no one seriously challenged this first and most durable of all standards until Martin Luther nailed his 95 theses to the door of the Castle Church of Wittenberg, almost 1200 years after Constantine’s bishops returned to their episcopal sees.

And so we see that the practice of standards development as we know it today has a very long and successful history indeed, although you might not have thought about it quite this way until now (*confess!*) Indeed, it must be admitted that this first great standards conference has been the most successful, if not the most acknowledged, of all to date. Why? Because the standard that Constantine’s bishops adopted by consensus came to be known as the *Creed of Nicene*. It has

been read in every Catholic (and Eastern Orthodox church) as part of every mass ever celebrated, in any church, anywhere, ever since – nearly 1700 years in all.

Rather an interesting, even *un-boring* standards story, if you stop to think about it. And all true besides.

So take *that*, Dan Brown.

Copyright 2009 Andrew Updegrove

Read more *Consider This...* entries at: <http://www.consortiuminfo.org/blog/>

