The Open Source Myth
Why Executives Should Think Twice about Making Company Source Code Public

Michael G. Wagner
Department of Computer Science and Engineering
Arizona State University
Tempe, Arizona 85287
Email: wagner@asu.edu

J. Randall "'Randy"' Jue
Etherton Law Group LLC
Tempe, Arizona 85282
Email: randy@ethertonlaw.com

***Abstract*— This article attempts to address the issue of whether open source software companies may be undermining their own competitive advantages by undervaluing the strategic importance of intellectual property. If this would be true, companies could risk their existence by following a potentially destructive business model. We will first take a look at the historic development of the open source movement. After taking a critical view on some issues important within the commercial domain we will address the question of whether open source is a viable option for companies operating in competitive environments and will furthermore present three alternative, hybrid licensing approaches.**

I. Introduction

Along with the Internet, the concept of open source software has been hailed as one of the biggest drivers of the electronic economy. Proponents of the concept not only describe it as a new and better way of developing software - they also assert the concept supports a variety of business models. The enthusiasm for the open source concept has lured companies into adopting open source wholeheartedly; however, lost in the euphoria is the unanswered question of whether it is indeed a commercially viable alternative to traditional software development. It appears obvious that there is an intrinsic difference between "software development model" and "business model." Yet the media and companies that have adopted the so-called open source model, if it exists, have blurred this distinction.

A. Competitive Strategy

Following the strategic thinking of Michael Porter [9], the fundamental rule for any business operating in a competitive environment is to obtain, develop and maintain sustainable competitive advantages. Within the software industry the most powerful source for competitive advantages is the creation and maintainence of intellectual property in the form of source code. In contrast, the proponents of open source development regard intellectual property as a retardant to software development, which ought to be abandoned in order to stimulate network effects. One could, of course, argue that the software industry is not fundamentally competitive; that it allows a certain amount of cooperation between companies in an effort to establish a common standard that benefits not one but a whole group of companies [12]. Nevertheless, on its most fundamental level, software developers have no other choice but to compete for the end consumer. Cooperation and standardization just define the arena in which competition takes place - they do not reduce the need for competitive strategies.

B. Open Source Model or Myth?

Some have attributed the dot-com crash partly to a failure by those companies to implement traditional ways of competing [10]. This article attempts to address the issue of whether open source software companies may be repeating the sins of the dot-coms - that these companies may be undermining their own competitive advantages by undervaluing the strategic importance of intellectual property. If this would be true, companies could risk their existence by following a potentially destructive business model, which motivates the question: Are we indeed dealing with an open source model or more with an open source myth? In the following we will first take a look at the historic development of the open source movement and proposed open source business models. After taking a critical view on some issues important within the commercial domain we will then address the question of whether or not open source is a viable option for companies operating in competitive environments.

II. A Brief History of Open Source Time

The open source software movement is built on the idea of freely distributing the "source code" of software encompassing all the lines of code or instructions, typically written in certain high level programming languages such as C++ or Java. The term "high level" refers to the fact that these languages are closer to natural language making it readable by any programmer. In a traditional software development environment, the source code of a company's software product is regarded as the company's intellectual property. The software product is distributed in object code, or binary form, which is only readable by a computer, while its source code is only released to others who are willing to pay for the right to see the code under strict licensing agreements.

According to the proponents of the open source concept, the "openness" of the approach results in several key advantages

over closed source development. First, by making the source code available to everyone, a multitude of programmers can help modify the code and fix bugs, or flaws, in the code. Under the closed proprietary model, a user must go through the company's customer service department, which in turn must report the problem to the corresponding software development unit. Second, because so many people are contributing to the code, resulting networking effects are expected to lower the cost of software development, which should translate into less expensive software. Third, because the code may be modified, users can customize the code to fit their needs.

A. Free Speech in the Bazaar

The open source software concept started as a political idea called "Free Software", which was popularized by Richard Stallman in 1984 when he formed the Free Software Foundation and its GNU Project [4]. Open source licensing is based on several key principles set forth in the Open Source Definition [2], which began as a policy document of the Debian GNU/Linux Distribution, an early Linux operating system. In its context, the meaning of "free" is non-proprietary, not non-commercial. According to Stallman we should "Think 'free speech,' not 'free beer.'"

In 1997, a proponent of free software, Eric Raymond, contacted a programmer Bruce Perens about the idea of open source. After numerous discussions, the two formed the Open Source Initiative, an organization for managing the open source campaign and its certification mark. Perens wrote the first draft of the Open Source Definition as "The Debian Free Software Guidelines" [4], and refined it using comments from Debian developers in a month-long e-mail conference in June 1997. If a license does not comply with the principles outlined in The Open Source Definition, its software cannot be labeled OSI Certified, a mark specifically governed by The Open Source Initiative [2].

In Raymond's view, which he published in his fundamental work "'The Cathedral and the Bazaar"' [11], open source development resembles "a great baabling bazaar of differing agendas and approaches" very different from the traditional "cathedral" approach used in proprietary developments. Indeed, it has to be noted that the open source movement promotes many concepts that have caused significant advanced in the way we look at software engineering. In some sense, open source introduced the notion of distributed software development [3].

B. A Definition of Terms

As described in [2], the Open Source Definition contains the following terms:

1) Redistribution Free - 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 the source code, and must allow distribution in source code as well as compile 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 fro 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 distribute 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 Field of Endeavor - The license must not restrict anyone form 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 Contaminate 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.

C. GNU Copyleft

Of the various open source licensing, one of the oldest and well-known licenses is the GNU General Public License ("GPL"). The GPL allows users to copy, modify, and distribute the software conditioned on the user's agreement to license all derivative versions under the same terms. The GPL has a total of 12 terms and conditions. To summarize, the GPL mandates

that users must agree to the following:

1) Users may not establish proprietary rights in the software;
2) Users must provide the source code to anyone to whom they give the object code;
3) Users must include notice in the software regarding the applicability of the GNU GPL; and
4) Users must accept the software without warranties of any kind.

The key element in the GPL is the GNU Copyleft principle, which requires all open source derived software to be open source as well. While Copyleft has achieved broad acceptance around the world, it has also been argued that it is at odds with some principles of the American economy [5], [6].

III. Open Source as a Business Model

It is important to point out that in this paper we do not address the issue of whether software developed via the open source approach is superior to software developed through a closed proprietary method. To the profit focused company, this is a secondary issue. Even if it were true that open source development produces superior results, the question remains whether profitability can be achieved when a company opens its product line to competitors. The open source business model generally requires that software be given away for free. To achieve a positive cash flow companies therefore have to focus on other sources of revenue, including product support and consultation.

A. Open Source Profitability

In ”Open Source As A Business Strategy” [4], Brian Behlendorf, a co-founder and a core member of the Apache group, analyzes how this profitability is achieved. Behlendorf describes a fictitious database company where 40 percent of the revenue comes from selling the software database product. The rest of the revenue comes from support, consulting work, rapid development tools, graphical administration tools, library of stored procedures/applications, and manuals. According to Behlendorf the loss of this 40 percent - which will happen when the company gives the software away for free - is recouped through the other sources of revenue. Specifically, Behlendorf argues that:

1) The company will have more ”freedom” to charge more for these other services than before because the customer is freed from the cost of the software, which ate up the bulk of the software package that was bought.
2) The ”free” software will double the amount of systems using the software.
3) Costs of software development should go down because motivated customers will likely fix bugs themselves and innovations in the software will occur because customers will contribute code to ensure that the software is maintained as a standard part of the overall distribution.
4) The availability of the source code will only marginally assist competitors who will compete to provide support for the software because your company - as the original developer of the software - will have brand-name recognition.

Behrendorf’s open source strategy essentially treats the software product as a very expensive ”loss leader”, which will tempt customers to acquire additional services. While this point of view seems convincing, it is not without problems. In a typical retail environment, a loss leader is only one article among many. In the open source environment in the other hand, the loss leader amounts to the entire product line. Moreover, Behrendorf’s analysis appears extremely speculative. In particular, it assumes that competitors will not be able to engage in active brand building to counteract the recognition of the original open source brand. Linux distributions such as Red Hat, SuSe, or Mandrake, however, have clearly shown that it is easily possible to build recognized and successful brand names on top of the original open source brand.

B. A Proof of Concept

The proponents of the open source concept typically point to the Linux operating system as proof of the concept’s economic viability. Linux is a version of the Unix operating system based on a kernel developed by Linus Torvalds of Finland along with many collaborators worldwide. Over time, Linux has attracted a cult-like following among programmers and systems developers who say it is a more secure, flexible and economic alternative to Microsoft’s industrial-strength operating system, Windows XP. Torvalds, who wrote Linux in 1991, when he was a student at the University of Helsinki, licensed it in a way that allows anyone to submit improved code and redistribute it at will. Since then, thousands of programmers have volunteered elaborate improvements of their own design.

Companies that sell the Linux operating system rely on making money from ancillary revenue sources. Since the software itself is given away for free - or may be obtained on the Internet for free - fees are earned by bundling additional software with the free software or by providing training or services. Although Red Hat, and other companies associated with Linux, raised huge sums of money from eager investors in the 1990s, it has not been proven that companies trafficking in open source software will achieve sufficient profitability. Even Red Hat, which is considered the leader in Linux-driven companies, admits in a recent filing with the Securities Exchange Commission (SEC) [1] that: ”Our open source software business model is unproven. We have not demonstrated the success of our open source business model, which give our customers the right to freely copy and distribute our software. No other company has built a successful open source business. Few open source software products have gained widespread commercial acceptance partly due to the lack of viable open source industry participants to offer adequate service and support on a long term basis.” And while this statement is certainly not surprising when given the legal implications of an SEC filing, it is strikingly similar to statements made by dot-com companies in the late 1990s.

C. IBM's Linux Strategy

Along with Red Hat, the open source software community's other favorite example of validation is IBM's adoption of the Linux operating system for much of its hardware. In early 2001 IBM announced that it will put Linux on all of its computer models, from wristwatch-computer prototypes to mainframes, hoping to combine the flexibility and low costs of open-source software with IBM's strengths in hardware and services. For example, IBM would offer its Global Services customers a cheaper alternative to licensed software, which the company claimed would lead to more consulting contracts and more hardware sales.

The media has portrayed this alliance between Linux and IBM as a tremendous victory for the open source community. When IBM announced its decision towards Linux, the company said it was doing more than choosing an operating system - that it was catching a wave of the future. It appeared as if the decision was entirely the fact that Linux is open source software; however, it is important to keep in mind that IBM is primarily a hardware vendor with a history of failures in developing a successful operating system. By outsourcing the operating system for IBM's proprietary hardware platforms to the Linux community, IBM made the decision to refocus on its core competency. In other words, IBM has not abandoned traditional business practices. It is still building its business around proprietary intellectual property. This is strikingly different from a software company like Red Hat, which is trying to build a business without proprietary intellectual property.

IV. A Critical View on Open Source

Unfortunately, the open source discussion is often led by beliefs rather than facts clouding advantages as well as disadvantaged. In the following we discuss five important issues that need to be considered when we examine open source licensing in a commercial environment. First, Linux is often used as a univeral example even though its success might not be a result of the open source license at all. Second, open source is often popularized mainly as a way to fight the monopolistic software giant Microsoft. Third, liability issues are usually neglected even though they are of fundamental importance in commerce. Fourth, the use of revolutionary language by the open source movement has to raise eyebrows in any case; and finally, the American Constitution provides no reason to argue that freedom and proprietary development are mutually exclusive concepts.

A. Linux Is No Universal eXample

Because the Linux operating system has achieved some success in the marketplace as an alternative to Microsoft's Windows, it has evolved into the role model of the open source software movement. However, it is questionable whether its success is solely due to its development as an open source project. Just as it would be an oversimplification if a medical researcher would be trying to narrow the cause of a disease to a single item of food without taking into consideration environment, genetics, and all the other food items consumed, the success of the Linux operating system is likely a much more complicated picture.

As noted above, Linux is a version of the Unix operating system. To use Linux in its purest form, a user must be adept with a complicated syntax of arcane commands. In dramatic contrast, the Windows operating system is built around a complex and feature-rich graphical interface that is perceived to be extremely user-friendly. The main advantage of Linux is that its architecture allows it to be significantly more stable, secure and flexible than the Windows operating system. This is an advantage for business applications that require reliability and not so much user friendliness. Only very recently, there has been a push towards more user-friendly Linux or Unix based user interfaces, such as Gnome or Mac OS X, tailored for the consumer market.

B. Fighting Microsoft

Another reason for the surge of Linux is a backlash against Microsoft, who dominates the software market in an almost monopolistic way. The company is often seen as an evil force that is trying to take over the world - a perception only reinforced by the federal government's anti-trust suit and its settlement. At the same time, many members of the open source community view the ”free software” idea as more than just a software development paradigm. To some extent it is a belief system. Thus, the competition between Linux and Windows operating systems is unnaturally elevated to the degree of an almost biblical confrontation.

Linux's success is also based on the fact that the open source software approach represents the most effective way to enter a market that is dominated by a single developer. One barrier to the entry of newcomers in the technological marketplace is a phenomenon known as the ”network effect.” Consumers and businesses dislike adapting to new systems, increasing switching costs and therefore establishing barriers to entry by newcomers. A similar situation can be found in the instant messaging market. When teenagers became hooked to America Online's Instant Messenger, AOL took the opportunity to seize the market to such an extent that even a large company like Microsoft still struggles to enter.

C. Never-Never Land

In the fairly tale Peter Pan, one of the major themes explored is the reluctance to grow up, to avoid responsibility. The tale describes a special place where kids can live and be happy without ever maturing into adulthood. Similarly, open source developers to some extent are attempting their own version of Never-Never Land by avoiding any responsibility for the software they create. This may be useful when a piece of software is based a recreational project passed among hackers. But when that piece of software evolves into a complex consumer product that will be distributed to millions of people, this risk avoidance strategy is unrealistic.

Most open source licenses are designed to deflect risk away from the software developers. As one open source advocate has stated: ”If free software authors lose the right to disclaim

all warranties and find themselves getting sued over the performance of the programs that they've written, they'll stop contributing free software to the world. It's to our advantage as users to help the author protect this right." Unfortunately, it is a basic business tenet that at the end of the day someone must be accountable when something goes wrong. The open source software approach is not foolproof and some open source software will produce defective products. And even if it is true that the open source developers will respond more quickly to bugs than a closed proprietary company, this will not rectify the damage suffered by consumers.

D. The Open Source Revolution

Unfairly or not, critics of the open source community belittle its members for the fervor with which they defend the open source movement. It cannot be denied that the open source community considers itself a somewhat political organization. The open source software community consistently borrows terminology from the American Revolution. One of the favorite mantras of the members of the open source community is "freedom." The members of the community brazenly describe the movement as the "Open Source Revolution." Bruce Perens has described The Open Source Definition as "the bill of rights for the computer user." Yet, despite all of this revolutionary rhetoric, the open source community rejects a notion embraced by the Framer of the United States Constitution - the value of intellectual property.

E. Freedom and Intellectual Property

At its core, intellectual property is about the incentive to create something new and original. In the Constitution, the Patent and Copyright Clause explicitly states the purpose of protecting intellectual property is "[t]o promote the Progress of Science and useful Arts." In Mazer v. Stein, 347 U.S. 201 (1954), the United States Supreme Court stated as follows: "'[t]he copyright law, like the patent statutes, makes reward to the owner a secondary consideration'. The economic philosophy behind the clause empowering Congress to grant patents and copyrights is the conviction that it is the best way to advance public welfare through the talents of authors and inventors in 'Science and useful Arts.'"

The reason for this is easily understood. The creation of something new - such as software, for example - requires tremendous amount of time and other costs to the programmer. A programmer will not invest time in creating new software unless his expected return from doing so exceeds the cost of doing so. The programmer must earn some form of profit from the endeavor. The open source community asserts that there exists a pool of hackers that will be drawn to open source projects for the glory of getting the credit of having contributed to the project. While there may be programmers who fit into this category, it is unrealistic to expect that enough hackers motivated only by glory will be make themselves available for open source projects. Furthermore, as the number of open source project increases, the benefit of being part in a development community diminishes making it less attractive for the average programmer.

The open source community correctly points out that there are costs that accompany the creation of intellectual property right. One cost is that the right to exclude others limits the diffusion of the innovation to others. Another cost is the holder of the intellectual property right may charged more than the marginal cost, which means that fewer consumers will be able to benefit from the innovation because of the prohibitive cost. However, the Framers obviously recognized these costs as well, which is why the intellectual property rights are limited in scope and duration. The framers of the constitution believed in freedom *and* intellectual property.

V. To Open or not to Open?

So, does it make sense to open a company's source code following the open source model? As in many cases, there is no universally applicable answer to this question. In a non-profit environment, open source is an excellent tool to circumvent licensing issues that might arise from the environment the organization is working in. Academics, for example, quite often use open source licensing to bypass the rather strict intellectual property rules enforced by many Universities.

From a commercial perspective, the main flaw of the GPL ("the pure open source licensing model") is that the Copyleft requires a company to require any code changes to be made available to the public including the company's competitors. This eviscerates an important way of obtaining a sustainable competitive advantage. Consequently, the GPL makes little sense for commercially released software if the company considers this software as one of its main assets. To achieve profitability in the software industry, a company must be able to build a business around proprietary intellectual property.

A. Three Alternative Licensing Strategies

If the pure open source licensing approach is not the answer, then what is an alternative solution? Is there a middle ground between the pure open source licensing model and closed proprietary model? This may be possible in three ways. One way is the multiple licensing approach. For example, the Berkeley Standard Distribution ("BSD") license allows a company to distribute modified software without disclosing the modifications. It also allows for the software to be incorporated into a combined software product that uses closed proprietary software.

A second way of facilitating a company's ability to build its business around proprietary intellectual property might be termed a "Delayed open source approach." Under this licensing model, royalties would be paid for limited number years to the open source developers with the stipulation that the software must be later released as open source software. This allows the company to recoup its investment in innovation. It somewhat resembles the patent approach.

A third way could be to create a patent pool. Under this model, various companies would join patent pools created by cross-licensing. The advantage for a company in joining a

patent pool would be to eliminate the costs of negotiating with several different companies for related technology that needs to be combined in order to produce a particular product.

VI. Conclusion

Open source has established itself as a very important tool in today's software industry; however, there is absolutely no reason to believe that open source has to be the only driving force in software commerce. Company executives should analyze very closely if the opening of source code undermines the strategic position of the company before they decide on licening through an open source model. At the same time, open source should not be treated as a threat, but rather as a mechanism capable of identifying and correcting problems within the software market. Open source developments are driven by consumers. A successful open source project is usually an indication that software companies are not properly responding to the needs of their customers.

References

[1] Form S-3, Registration Statement under The Securities Act of 1933, filed by Red Hat, Inc. with the Securities and Exchange Commission on July 20, 2001. [Online]. Available: http://www.sec.gov/Archives/edgar/data/1087423/000102140801503660/ds3mef.txt.

[2] (2002) The Open Source Initiative (OSI) website. [Online]. Available: http://www.opensource.org.

[3] C. Davis and C. Bayrak, "Open Source Development and the World Wide Web: A Certain Tension," *ACM Software Engineering Notes*, vol. 27, no. 5, pp. 93-97, 2002.

[4] C. DiBona, S. Ockman, and M. Stone (eds.), *Open Sources: Voices from the open source revolution*. O'Reilly, 1999.

[5] R. Gomulkiewicz, "The License is the Product: Comments on the Promise of Article 2B for Software and Information Licensing," 13 *Berkeley Technology Law Journal* 891, 1998.

[6] R. Gomulkiewicz. "How Copyleft Uses License Rights To Succeed In The Open Source Software Revolution And The Implications For Article 2B," 36 *Houston Law Review* 179, 1999.

[7] M. A. Lemley, "The Economics of Improvement in Intellectual Property Law," 75 *Texas Law Review* 989, 1997

[8] J. Lerner and J. Tirole, "Some Simple Economics of Open Source," *Journal of Industrial Economics*, vol. 52, pp.197-234, 2002.

[9] M. E. Porter, *Competitive Strategy: Techniques for Analyzing Industries and Competitors*. Free Press, 1998.

[10] M. E. Porter, "Strategy and the Internet," *Harvard Business Review*, March 2001.

[11] Eric Raymond (1997) The Cathedral and the Bazaar. [Online]. Available: http://www.tuxedo.org/ esr/writings/cathedral-bazaar/.

[12] D. Tapscott, "Rethinking Strategy in a Networked World," *Strategy+Business*, vol. 24, 2001.