Open Source Software Development

Thesis for Senior Honors at Brandeis University

Robert L. Greenberg*

May 9, 2003

This thesis is written for graduation honors in the Department of Economics at Brandeis University. In this thesis I hope to explain from an economic perspective how Open Source Software Development works. In doing so I examined the current literature available and historical examples and I developed case studies on instances in the Open Source and business communities where this form of development has become prevalent.

All errors are that of the author.

* Undergraduate at Brandeis University in the Dept. of Economics. Special thanks go to Prof. Anne P. Carter of Brandeis University for her guidance and advice.

Table of Contents and Summary

I. Introduction

Open Source Software (OSS) Development has come of age. Weak intellectual property protection has allowed for the development of high quality software. Herein I hope to explain this process through economic theory. I examine the current literature available on the topic, develop cases looking at the state of various OSS projects and firms in their reaction to Open Source as a model of software development, and examine historical instances of free knowledge trading. Through this examination I hope to answer the ultimate question of "How does Open Source Software Development work in the context of current economic theory?"

II. Definitions of Terms in Open Source Development

Intellectual Property

Intellectual property (IP) is an artificial legal construct. IP can be thought of as property rights extended to the creators of information. Information, in this context would be anything that can be expressed in digital form. This could include text, music, video, and executable files. Any form of information could be reproduced with no degradation in quality from copy to copy. The giving away of information does not leave the original holder with any less information, although it may leave the original holder worse off than before. This is how IP differs from physical property and why the idea of IP is "artificial." Because this information can be passed along without compensating the original authors, the United States Constitution charges Congress with creating laws that allow the original authors rights over the information they create.

The debate over IP rights comes from the founders' wish to provide incentives to inventors, who might profit from monopoly control over a new idea, while protecting the public's ability to benefit from the new idea. In the context of this thesis, the discussion of IP rights will be concentrated on the rights of producers of computer code over that code. In Open Source, the code authors forfeit any rights to the code in order to reap greater benefits. This seems to be counterintuitive as IP rights are thought of as a necessary precondition for the authorship of information. Herein I will prove that strong IP rights are not a necessary precondition to software development in all cases.

License

Holders of intellectual property sell, or license, their property for certain uses. Purchasers of books, for instance, are actually purchasing the right to read the book and resell the book at a later date. There are other rights such as copying a few pages or citing the book for academic research which come along with a book purchase. Software works differently. Since, unlike a physical book, software can be copied virtually costlessly, purchasing software is actually purchasing a specific license to use that software. Different software packages use different forms of licensing schemes. For instance, one could purchase one copy of Microsoft Windows at a computer store and make *n* copies of the software and distribute them. However, the actual purchase of the box means that the individual is purchasing the license to use the software on only one computer. Open Source Software is licensed under a number of differing licensing schemes. The stipulations on use differ from license to license.

Free Software

> The word 'free' has two different meanings in the English language, one suggesting a price of zero and one related to the idea of liberty. Richard Stallman, whose Free Software Foundation has long championed the term, says, "Think free speech, not free beer" but the ambiguity of the term has nevertheless created serious problems—especially since most free software is also distributed free of charge (Raymond 2001, 176).

The philosophy of Free Software is not about cost. It is about the freedom of the source code as freedom of speech. The theory behind Free Software is that when the code is available to read, only then can innovation truly occur. The Free Software Foundation (FSF) puts it as such:

> Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:
>
> - The freedom to run the program, for any purpose (freedom 0).
> - The freedom to study how the program works, and adapt it to your needs (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 to the public, so that the whole community benefits. (freedom 3). Access to the source code is a precondition for this. (FSF 2002)

In most commercial software, the code is closed and no one knows nor can know how the tools that they use everyday work. Few people know the recipe for Coca-Cola, but few businesses (other than potential competitors) would benefit from that recipe. Software is integral to the day to day operations of countless firms and in the lives of individuals worldwide. That is, millions of people use products like Microsoft Windows, Office, Internet Explorer, and so on, but none of those people can find out how they work. This kind of closes system is a novel concept outside the world of proprietary software. Physical products can often be taken apart, played with, and learned from. In

many instances, these products are patented so that they can be learned from but not copied by potential competitors. Software is fundamentally different in this respect. It is something that does not exist in a physical form. One analogy is that of literature. While books are physical, as are computer media like CD-ROMs and floppy disks, the work itself is just words. While a copyright applies to literature, these works are still available for many uses. If these same copyright laws are to apply to software, then code would presumably be available to be read but not to be copied. However, code for most commercially available software is closed and cannot be read by human beings. Similarly patents offer protection for owners but still allow those who wish to learn from anything patented can still read the patent, learn from the patent, and even make alterations through cross licensing.

The theory of Free Software is that people should be able to read from the source and do whatever one would wish to do with that software. Just as one can read, work, and learn from literature, Free Software advocates believe that everyone should be able to do the same things with software code. In this light, the FSF founded the GNU project. GNU recursively stands for "GNU's Not Unix," and was supposed to be a totally free alternative for proprietary Unix operating systems. The FSF had developed an entire operating system save one important part, the kernel or core of the operating system. Linux has since come along to fill that niche and the FSF calls the operating system with the Linux kernel the GNU/Linux operating system. (FSF 2002) The success of Linux has done much to facilitate the successes of Open Source.

Free Software vs. Open Source Software

This paper is fundamentally about Open Source Software (OSS) and Open Source Software development. Open Source Software is Free Software. The product is the same, but the philosophy is different. McGowan quotes Richard Stallman of the FSF as saying, "'The Free Software movement and the Open Source movement are like two political parties within our community. ... We disagree on basic principles, but agree on most practical recommendations' and 'work together on many specific problems'" (2000, 22). While the Free Software movement focuses on the free speech and philosophical arguments for source code availability, the Open Source movement is more about the economic benefits and practicality of having the source code available. Both support the idea of having the source code availability. The Free Software community has more than just a streak of Marxist-style idealism, while Open Source supporters have a more corporate friendly attitude. The FSF believes that closed source and copyrights are inherently unethical. Rather than focus on the ethical issues involved in intellectual property protections, this thesis will rather argue from the practical, Open Source standpoint.

It is also interesting to note that the Open Source movement grew out of the need to make Free Software, which tends to be of high quality, friendly to the corporate world. Eric S. Raymond is credited with being the first to write about OSS from a practical standpoint. His main treatise, the title essay in *The Cathedral and the Bazaar*, is all about these pragmatic benefits of Open Source software. The ideas that the more eyeballs look at the code, the more bugs that will be found, that companies that need solutions can immediately fix their code because they have it in front of them, that is they need not wait

for some other company to come and fix the code, that new features are being added by other individuals, groups, and firms to the software for free, etc. were very appealing and had gotten support from more traditional software firms. Raymond and other "key people in the Silicon valley and national Linux community" decided to develop a long term strategy to promote the benefits of this kind of software development (Raymond 2001, 175).

> It was easy to see the outlines of the strategy. We needed to take the pragmatic arguments I had pioneered in *The Cathedral and the Bazaar*, develop them further, and push them hard, in public. …
>
> The real conceptual breakthrough, though, was admitting to ourselves that what we needed to mount was in effect a *marketing campaign*—and that it would require marketing techniques (spin, image-building, and rebranding) to make it work. … Hence the term 'open source' (Raymond 2001, 175).

Pragmatism and practicality versus philosophy and fervor are the differences between the Open Source and Free Software movements. It is the same product with friendlier packaging. This is also where the split between the names Linux and GNU/Linux. The Open Source community prefers the name "Linux" in large part because it purposefully does not mention the FSF's GNU project, which would be a constant reminder of the less business friendly FSF's Free Software philosophy.

Linux

There is no "Linux Operating System." Linux is the name of the kernel of the operating system. The kernel itself is the underlying code on top of which everything else runs. Pointing this out is not, of course, to downplay the achievements made by Linus Torvalds and the many contributors in the development of the Linux kernel, but on the contrary, the story of its development is one of the great stories of computing history. (various sources)

The proper nomenclature of the operating system is a matter of some debate. Most people simply refer to the combination of the programs that lie on top of the kernel and the kernel itself as "Linux." Since most of the programs that run on top of the Linux kernel are Free Software from the FSF's GNU project, it is often referred to as the GNU/Linux operating system. The FSF explains it as such, "Variants of the GNU operating system, which use the kernel Linux, are now widely used; though these systems are often referred to as 'Linux', they are more accurately called GNU/Linux systems" (2002). Linux distributions are composed of more than just the GNU project and the Linux kernel. For purposes of this paper, I will refer to the operating system simply as Linux.

Distributions of Linux

Not all distributions of Linux are created equal. Different organizations and firms make their own combinations of the Linux kernel and other programs. Some of these distributions are Red Hat Linux, Debian GNU/Linux, Slackware Linux, and Mandrake Linux.

The differences between these distributions go deeper than just the combination of programs included. Certain distributions are configured in such a way that they work on different platforms than others. Some distributions run on Intel x86 computers, others on Alpha or PowerPC platforms. Others use different packaging, or upgrade distribution, solutions.

These different distributions are also developed by different organizations with different goals. Red Hat is a United States based commercial firm which is profiled in

one of the case studies included herein. Mandrake Linux is also distributed by a commercial firm based in France. Debian GNU/Linux is developed by a not-for-profit concern, and has a longer development cycle due to its focus on stability rather than on cutting edge features. No one simply runs Linux.

III. Examination of Open Source Licenses

A software license is a term of use agreement between the software user and developer. Use of the software is indicates that a user accepts the terms of the license agreement, and is legally bound to the terms of the agreement. There are many licenses available that are considered "Open Source" licenses. The ones I will focus on here are the General Public License (GPL) and the Berkeley Software Distribution (BSD) license. Software released on the GPL and the BSD licenses are both examples of free software and both release both the software and the code for free in both senses of the word. However, the stipulations in these licenses make the products released under them decidedly different.

While both licenses claim to be the most free, the BSD license is arguably the freer of the two. With a couple of minor caveats, the BSD license gives away their code for free. Unlike the GPL's "copylefting," a user of BSD software is allowed to do whatever he or she wants. That includes taking the code and incorporating it into commercial, closed source products. Apple's latest operating system, Mac OS X, includes BSD code and has been released as open source, but under a different, more restrictive license (Apple Computer, Inc. 2002, 2001). Microsoft Windows NT and its descendants, Microsoft Windows 2000 and XP, include BSD code as well.

The FSF has come up with a unique concept called copylefting. The GPL is a "socially and legally complex document that, as of this date, *is untested*" (McGowan 2000, 13; emphasis mine). Essentially the GPL ensures that the code can never be taken private. That is to say, no private firm can take Linux code and use it to create closed software. All code licensed under the GPL is forever open, as are the programs written

that use that code. The GPL states, “You must cause any work that you distribute or publish, that in whole or part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this license” (FSF 1991). The effect of a license such as this is that in a large project such as the Linux kernel, “as a practical matter, each contributing programmer would have to agree to privatize the code if it was to be taken private in its most current and complete form” (McGowan 2000, 20). In a project such as the Linux kernel, with all its contributors scattered all over the world, doing so would be, from a practical standpoint, impossible. The Linux kernel will be now and forever free. They use the power of copyright law, which is typically the power to exclude, and use it to keep code free and restrict the power to exclude. This is where the term “copyleft,” favored by the FSF, originates. This is the power behind the GPL. This is also from where the criticisms of the GPL arise.

The problem behind the GPL is that it causes all its derivatives to be free. If a programmer were to “borrow” some GPL code and use it in a proprietary program, she would be in violation of the GPL and would be sued to open her code. Because using even snippets of code causes the whole program to fall under the terms of the GPL, many groups are against this. This includes, perhaps unsurprisingly, Microsoft, which officially says the following:

- Some open source licenses are viral, that is, they require that all derivative works be licensed on the same terms as the original program. These licenses are described as viral because they "infect" derivative programs. Viral licenses vary in how infectious they are, depending on how they define which programs are derivative works. However, one of the dominant open source license—the GPL—is the most infectious. It attempts to subject any work that includes GPL-licensed code to the GPL. Thus, if a government or business uses even a few lines of GPL-licensed code in a program, and then re-

> distributes that program to others, it would be required to provide the program under the GPL. And, under the GPL, the recipient must be given access to the source code and the freedom to redistribute the program on a royalty-free basis.
>
> - Open source licenses that are non-viral, on the other hand, permit software developers to integrate the licensed software and its source code into new products, often with much less significant restrictions. A prominent example of this type of license is the Berkeley Software Distribution (BSD) license. The BSD license allows programmers to use, modify, and redistribute the source code and binary code of the original software program, with or without modification. Moreover, programs containing code subject to the BSD license are subject to only limited obligations imposed by that license. This type of license gives users freedom to incorporate their own changes and redistribute them, without requiring them to publish the new source code or allow royalty-free redistribution. (Microsoft 2003a)

Microsoft's objections have validity but are often dismissed as typical of Microsoft's corporate tactics. Many pundits point to the infamous Halloween Documents that leaked from Microsoft in October 1998. The Halloween Documents are internal Microsoft memoranda. The first one was believed to have been first written on October 31, and as such they were dubbed the Halloween Documents by the media. In these documents the need to portray the GPL as viral and to attack the license rather than attack GPL protected software is listed as a potential strategy (Open Source Initiative 2003). While these claims on the part of Microsoft may be part of its corporate strategy, these criticisms are not without merit. If someone were to quote a passage from a book, she would need to cite the author, but her final product would not need to fall under any licensing agreement from the original work. Because the original work falls under the GPL, all derivatives also fall under the license and in that way it is considered viral as it spreads from program to program "infecting" all code it touches. Microsoft critics do accurately point out that since almost all Microsoft code is closed, there is not even the opportunity to even learn from Microsoft's code. Recently Microsoft appears to be

opening up portions of its code, as is discussed in further detail in the Microsoft Shared Source case study.

IV. Current Theory on Open Source Software Development

The "Comedy of the Commons" (Raymond 2001, 154)

Garret Hardin's concept of the tragedy of the commons has become engrained in economic thinking. The idea that a free resource lacks sustainability has been proven throughout history with the example of the English commons, clean air, and fish in the open seas. Things are different today when we are dealing with the ephemeral. Unlike grass, air, and fish, information does not necessarily lose its value the more people that use it. Put simply, while fishing the seas clean deprives everyone else of fish, having a copy of someone's software does not leave her any worse off. In fact, in the case of Open Source, it can often leave users better off. This derives from the positive feedback loop and network effects that can develop around the software.

There are actual concrete costs in the development of software. People generally do not work without some form of compensation and firms that employ programmers expect to get benefits from the code that their developers create. "In a complex setting, the self-developer is hardly distinguishable from the commercial software firm … For this reason, I do not treat self-development separately" (Bessen 2001, 20). In terms of the economic analysis at this point, firms and individual developers are one and the same. Very often firms will assign or allow certain individuals to work on OSS projects, and assigning the credit to the individual or firm is difficult and ultimately immaterial for purposes of this paper. This is because firms and individuals make similar cost-benefit analyses, and conclude whether or not to develop. In this case, the answer to how individuals and firms can benefit from letting go of their intellectual property lies in a rethinking of how the software industry works.

"Part of the answer [to the question of how Open Source works] lies in the fact that people don't merely need solutions, they need solutions *on time*" (Raymond 2001, 125; emphasis author's). What Raymond means is that people cannot necessarily wait for a vendor to provide a solution to whatever problem they may be having. Finding a bug in a Microsoft product, such as Internet Information Server (IIS), is a fundamentally different experience than finding one in a comparable open source product like Apache would be. The difference lies in that finding a bug in IIS, would require Microsoft to fix it. This could take an indeterminable amount of time. In an Open Source product, it can be fixed *now*. It is not just important that the solution be had, but if the problem is important enough, the problem can be solved immediately. "If the payoff from fixing a bug or adding a feature is sufficient to any potential contributor, that person will dive in and do it (at which point the fact that everyone else is a free rider is irrelevant)" (ibid). I believe that free riding may be in fact be a problem for Open Source Software development as is discussed in the following section.

This explains why a developer would work on an Open Source project, but does not explain why these developers give their code away freely. Why there may be little to no costs involved in giving the code away, there does not at first appear to be an impetus to do such a thing. Developers can actually profit from giving away code freely.

> Suppose I write a fix for an irritating bug, and suppose many people realize the fix has money value; how do I collect from all those people? … It may be more to the point that this value is not merely hard to capture, in the general case it's hard to even assign. … It would take a superbeing, both able to evaluate the functional worth of patches and trusted to set prices accordingly, to lubricate trade. … Unfortunately, there's a serious superbeing shortage, so patch author J. Random Hacker is left with two choices: sit on the code, or throw it into the pool for free. (Raymond 2001, 126)

The idea that firms give away intellectual property that they could not profit from is nothing new. It has existed for many years and has been known to profit in the long term. Harhoff, Henkel, and von Hippel (2000) give us "four incentives which could induce user-innovators to freely reveal their innovations:" (1)

> a) *Inducing manufacturer improvements*
>
> By freely revealing an innovative product or process, a user makes it possible for manufacturers to adopt that innovation and improve upon it. …
>
> b) *Setting a standard advantageous to the user innovator*
>
> By freely revealing an innovation, a user makes it possible for other users to adopt it as their solution as well. This adoption as a "standard" can be beneficial to the user-innovator if his innovation contains imbedded features particularly beneficial to that user … Note that being first to reveal a given type of innovation increases a user's chances of having *his* innovation widely adopted, other things being equal.
>
> c) *Reciprocity and reputation effects*
>
> By revealing an innovation, the innovator may create or discharge "generalized reciprocity" obligations … He may also reap a gain in reputation
>
> d) *Low rivalry conditions*
>
> When competition between users is low … the revealing user does not suffer as a consequence of the advantages to the other users. This is not so much of an incentive as *absence of a disincentive*. (ibid, 2)

Inducing of manufacturer improvements is hard to translate directly onto a paradigm that often has no manufacturer in the traditional sense. A manufacturer could be considered any developer for a given platform. A parallel that could be drawn is with the Linux kernel. The very idea that it is constantly improving would lead one to improve it further if she found a bug. Fixing one bug could cause others to find further bugs and a user who knows that further improvements are coming along would be more likely to contribute to such a project.

Another example could be that of Fetchmail, an open source mail client for Unix machines. Raymond describes how he started writing patches for and eventually took

over an existing project and each time more patches came in and Fetchmail became a powerful and popular piece of software thanks in large part to the continuing innovations (Raymond 2001, 19-63). Still another could be the Beowulf supercomputer clustering project. The Beowulf project started at the NASA Goddard Space Flight Center because of their need for supercomputing abilities for calculations. Beowulf clusters are clusters of computers using commodity equipment, such as x86 based computers and using special software to have these computers run as one computer for various applications. (NASA 2003) NASA had neither the resources nor desire to do all the development for the project. By using Open Source, NASA was able to take advantage of the interest in the project from universities and other organizations in need of low cost supercomputers. The better that these cheap supercomputers ran, the more institutions adopted them and user-developers further improved the speed and stability of these clusters.

This is a positive feedback loop. In essence each time a contributor adds a new feature or fixes an existing problem; it causes more features and fixes to come down the pike. The better the software becomes, the more users would become inclined to use the software. The likelihood that one such user would improve it further increases with the greater number of users.

Setting a standard advantageous to the user is also an important concept. Phenomena like the network effect and positive feedback loops, which derive from having a single standard, contribute greatly to the private support of Linux. IBM benefits from having an open standard like Linux or Apache. Accepting and promoting these standards allows IBM to focus more on improving its hardware line. Helping to set the software paradigm gives IBM an advantage in sale of its products as firms often prefer to

run their infrastructures on a single standard. This gives a firm, such as IBM, a jump start on the competition in these fields as it innovates and standardizes.

There is another advantage in setting a standard. In fixing a bug, for instance, there are inherent costs in not distributing the patch. In Open Source, one of the fundamental philosophies is that of "release early, release often" (Raymond 2001, 28). This leads to new versions of a user's software to come out on a frequent basis. Each time that the new software is released, the user would have to re-integrate the code with the new release. This is a costly practice. Each time, the user would have to go back through the code and is likely going to have to edit the patch in order to make sure that the program works and that the previous problems are fixed all over again each time (ibid, 126). This could be thought of as the equivalent of replacing a washer somewhere in the middle of all the pipe works in a building each time the water filter is replaced. While necessary, it is time consuming, and it is often difficult to do. To avoid these costs, developers have an incentive to freely release their patches and developments.

Even non-contributing users can have an effect as well. There is the concept of evangelization and users can contribute in other ways. Users who do not contribute to the development of Open Source projects contribute by simply using the software. One could argue that by not contributing money to closed source software projects, i.e. not purchasing products like Microsoft Windows, one would be giving the free alternatives a competitive advantage. Users of OSS are also its biggest fans. A user of Open Source is likely to "spread the good word" of Open Source and convince others to use OSS as well. Many of these other users may contribute donations to organizations like the FSF or could hire firms such as IBM or Red Hat to assist in using open source software at an

individual or enterprise level. Furthermore, individual users can often find and report bugs back to the developer pool. These users may not be capable of fixing these issues, but their finding and reporting of them does help get bugs fixed all the more quickly. These may all be indirect benefits of otherwise free riding users, but as the costs of redistributing the software is minimal, these benefits outweigh the costs.

Reciprocity and reputation effects have become the basis for various business plans within the Open Source community. Red Hat's model is largely based on the reputation effects that arise from the production of high quality OSS solutions. Its distributions of Linux are a form of free advertising and through this it hopes to garner business and ultimately turn into a profitable consulting firm. Red Hat is a particularly interesting case and presents a novel business plan. Red Hat is discussed in further length in a devoted case study.

Lower rivalry conditions exist in Open Source than do in other forms of know-how trading. Because IT has become pervasive in all forms of business, fixing a problem in software not only affects the developing firm, but can help all businesses at large. This is different than a steel company that improves on its smelting technology. How this differs is that in a steel company, any improvement that is shared is shared directly with the developer's immediate competitors as well. There is no necessary competition between developers in the Open Source community. An improvement made to a web server at an investment bank will help a smaller web services company halfway across the world. There is no competition whatsoever between these two firms. While there are instances where two firms compete with much at stake, the rivalry conditions can be considered to be lower. These firms would not be competing directly on the quality of

the software that is run but can use the advantages of Open Source to shift this to other areas. IBM uses Linux and Apache to shift the focus to their hardware and consulting services, which may put it at an advantage against competitors such as Sun Microsystems. Also we can look back at a firm such as Coca-Cola. Coca-Cola could hypothetically release its improvements to its software freely knowing full well that it could assist other soft drink producers. The benefits accrued by releasing these improvements to the community at large outweigh the pitfalls of helping their competition. These pitfalls would be smaller than if Coca-Cola were to release its secret formula. The large size of the Open Source community allows for the rivalry conditions affecting any single firm to be comparatively small.

Economic Modeling of Open Source Software Development

Economists have recently taken up the study of OSS projects and bring to light some interesting analysis of when and how these projects can succeed. The prevalent theory is that open source is most successful when dealing with software that directly affects a developer community. "Many open source advocates argue that open source software tends to be geared to the more sophisticated users" (Lerner and Tirole 2000, 9). Linux and Apache are examples of this. Linux and Apache are software used by generally experienced users who run Unix and host web pages from servers.

> Certainly, the greatest diffusion of open source projects appears to be in settings where the end users are sophisticated, such as the Apache server installed by systems administrators. In these cases, users are apparently more willing to tolerate the lack of detailed documentation or easy-to-understand user interfaces in exchange for the cost savings and the possibility of modifying the source code themselves. (ibid)

Counterexamples, like the GNOME and KDE projects, which put a graphical front end on top of a Linux system, do exist. Still, these types of projects tend not to be very mature from an ease of use perspective, and are the exception to the rule. Most sophisticated users do not need easy to use graphical user interfaces (GUIs) and thus lack the motivation to develop GUIs (ibid, 26).

However, it is important to ask: what does motivate developers to work on open source projects? In simple economic terms, it is when "the net benefit is equal to the immediate payoff (current benefit minus current cost) plus the delayed payoff (delayed benefit minus delayed cost)" (ibid, 20). This is Raymond's argument of how software is often not just needed, but needed immediately. Lerner and Tirole go on to explain how the concept of opportunity cost comes into play. They explain how the cost of the downtime for an individual may be greater than the cost of working on the project. The opportunity costs of not working directly on the project at hand can be countered by the benefits accrued by having superiorly working software. "The programmer, when fixing a bug or customizing an open source program, may actually improve rather than reduce her performance in the mission endowed upon her by her employer. This is particularly relevant for system administrators looking for specific solutions for their company" (ibid). When the benefits outweigh the costs, the software gets improved. How often this occurs is still a matter of debate.

> Some open source projects have a greater number of potential developers in the community than others do. Reasons for this include differing awareness about projects and heterogeneity in the underlying value and cost distribution. The number of users in the community influences the equilibrium probability of development, the amount of redundant development effort, and social welfare more generally. …
>
> Suppose that the number of user-developers increases. If individuals continued to use their original threshold rule, then clearly development would be more likely.

> However when more individuals are present, the incentive to free ride is raised, and any individual will be less likely to develop the application herself in equilibrium. Whether the overall probability of development falls or rises as a result of including more agents is therefore ambiguous. (Johnson 2001, 9-10).

This does affect the calculus. When there are only a few developers working on a project, it does make sense for development to occur. The idea is that: If I don't do it, no one else will. This changes when the number of developers increases and the possibility of someone else correcting the problem also increases. This is the essential free rider problem in the OSS community. Johnson ultimately concludes that as the number of developers rises, so does the amount of development. "In equilibrium, higher values of *n* [the number of user-developers] deliver higher option values of doing nothing, without lowering the return on innovation" (ibid, 11).

Ultimately, the theory boils down most simply to a handful fundamental concepts. These ideas are immediacy, efficacy, and economy. A user capable of development considering her options would have to think about these three issues and ask a series of questions.

- **Do I need this solution immediately?** In order to justify development, a user-developer needs to be able time optimal solution for the issue at hand.
- **Will this be the best solution?** When alternatives exist to solve whatever issue is at hand, a user must examine which alternative solves this best. A perfect solution tomorrow may be inferior to a copasetic solution today.
- **What costs are involved in self-development?** If the cost of developing a solution to whatever issue a user is faced with exceeds the benefit that would arise from such solution, the development would not occur.

If these three conditions are met, user-development can occur. These conditions may not occur that often for any given user. Given the large number of individuals that use OSS solutions, the ease of information transfer which is facilitated by the Internet, and the

benefits which accrue from development, Open Source Software has become a viable form of software development.

Examination of Historical Correlations to Open Source

The idea that revealing information leads to greater innovation is not unique to the field of software development. There are historical examples of free release of innovations leading to benefits to the developers. Weak intellectual property protection has often lead to greater innovation than strong protection has allowed.

One recent paper has pointed to examples of information sharing in nineteenth century technology as an analogy to modern Open Source development methods. Nuvolari points to examples of "collective invention" in the Cleveland blast furnaces and Cornish mine engines in the UK (2003). More recent examples could be the know-how trading found in modern "mini-mills" in the United States. These smaller steel companies have found that through cooperation they have become increasingly profitable even as older steel manufacturers have fallen on hard times.

Another such example of this is in the development of copper interconnections for chip design made by IBM. The development of the use of copper would give the semiconductors produced by IBM a major advantage over those developed by its competitors.

> IBM was first to develop a process to manufacture semiconductors that incorporated copper interconnections among circuit elements instead of the traditionally-used aluminum ones. This innovation provided a major improvement to semiconductor performance, and on the face of it, it would have paid IBM to not reveal its process to others. …
>
> IBM freely revealed information about its innovation because it needed equipment to implement the process on a production scale. Detailed design and production of such equipment required the combining of information held by semiconductor equipment suppliers and IBM. Since development of novel

> process equipment is a very expensive matter in the semiconductor field, suppliers would only be willing to build equipment that could potentially be sold to the entire market. … Thus, IBM was motivated to "openly reveal" its innovation incentives. (Harhoff, et. al. 2000, 3-4)

Another historical example is the basic transistor. "If ever an innovative firm could have gathered large monopoly rents from its patents, Bell Laboratories could have done so from its basic transistor patents" (Bessen and Maskin 1999, 12). Bell Labs chose to give away this freely. The explanation for this comes from the vice president of electronic component development at the time:

> We realized that if this thing [the transistor] was as big as we thought, we couldn't keep it to ourselves and we couldn't make all the technical contributions. It was to our interest to spread it around. If you cast your bread on the water, sometimes it comes back angel food cake. (Bessen and Maskin 1999, 12; quoting another secondary source)

This analysis can be extended to the field of open source. The users are motivated to release the software because of the additional costs of not releasing it and because users believe that "to have their improvement incorporated into the standard version of the open-source software that is generally distributed by the volunteer open source user organization because it will then be updated and maintained without further effort on the innovator's part. This volunteer organization is the functional equivalent of a manufacturer with respect to incentive" (Harhoff, et. al. 2000, 6). Essentially the aggregate volunteer movement acts much the way a developing firm would as the incentive for innovation is equivalent.

In fact, with complex goods, free information can often lead to greater overall innovation than can the older model of patenting. James Bessen takes an interesting look at the idea of software patenting and applies it to both commercial and open source software projects. "Taken together, open source developers and commercial firms with copyright and trade secret protection provide a greater range of use-products than a

commercial firm with a patent monopoly. … For sufficiently complex products, a regime with user licensing, copyright and trade secrecy for software is preferable to a regime with software patents because the former permits greater provision" (Bessen 2001, 22). In another paper of his, he says that in cases such as software, "imitation becomes a *spur* to innovation, while strong patents become an *impediment*" (Bessen and Maskin 1999, 3; emphasis authors').

This overall improvement in the rate of innovation benefits developers who would be unable to otherwise profit from their innovations. Most firms do not have the capability of capitalizing on marginal improvements made to OSS projects. As such, these firms would gain the most benefit by having this weak protection. This is because a greater gain is had by taking advantage of further innovation than by attempting to sell the marginal innovations. For complex products, including software and the historical examples technological know-how trading, the benefits of the weak protection outweigh the benefits of strong protection.

I am inclined to agree with his conclusion, but I take issue with the methodology by which he came to this conclusion. Most software products are not protected by patents. Furthermore, most of these patented software products are quite specialized.

> Indeed, pure software companies as a whole have not applied for many patents. … In fact, the largest software patentees are in the computer hardware and telecommunications industries—industries which sell software products and also incorporate software in hardware products. … The top ranked pure software firm in 1995 was Microsoft (rank 24[th] in number of software patents]) with 39 software patents. (Bessen and Maskin 1999, 17)

The lack of an increase in R&D spending relative to the amount of patents, does not necessarily correlate with the lack of innovation in the software field. Telecommunications firms may in fact patent the software in order to further protect their

mixed hardware/software products. Strong protection for the software innovations may translate into strong protection for the hardware innovations for these firms. Furthermore, the reduction in R&D spending may also be a result in shifts in investing to other markets, including that of the pure software, and may not be necessarily a result of strong protection. Nonetheless, there is much evidence that supports the idea that open standards benefit innovation more than closed systems can.

Bessen does give us an explanation of the success of OSS projects by saying, "When products are complex and need to be debugged, markets do not work so well" (Bessen 2001, 23). The typical free market solution to the problems presented by complex, important projects like operating systems may in fact be best solved by Open Source methods.

The phenomenon of Open Source is still unique even in this historical context. The universality and ubiquity of software use in daily life means that this phenomenon is larger than the historical corollaries drawn. The fact that this is development of intellectual property is its own end is also different. In the historical cases, the information traded was for development of physical objects such as furnaces and microprocessors. In this instance, the final product is just as ephemeral as is the design for the product. Most of all, the major fundamental difference is in the development of the Internet. The Internet has facilitated information exchange on a scale never before seen in all of history. These phenomena have combined to catapult Open Source Software development into a new stratum of know-how trading.

Economic Problems with Open Source Software Development

Open Source development is not without its own problems. There are several problems with the assumptions that are made in the analysis of open source, and there are inherent problems with the development of any product that is free.

The problem in the assumptions arises from the assumption that if more and more people use these projects, more and more developers will work on them. This is not necessarily true. Bessen, in his conclusions says,

> I assumed that all consumers are potential developers. In reality, however, the actual customers for commercial software include both individuals and firms, larger and small. While large firms are usually potential developers as well as consumers, small firms and individuals are often not. (Bessen 2001, 23)

This is a large caveat in the theory of Open Source. The idea that many users are not developers brings in new complexities in these theories that often go unthought-of by writers on the subject. While the costs of free riding may be low, the idea that not every user is a developer changes the number of people that we may be thinking about. In much of the literature regarding OSS projects, we speak of a user-developer rather than a typical user. It is not safe to assume that every user of these programs has the ability to accurately find and debug bugs. In fact, the base of developers is, in actuality, much smaller than open source/free software evangelizers would have you believe. While the number of users of programs such as Linux and Apache is quite high, the number of those who actually develop the software is much lower. Evidence points to this fact.

> The distribution of contributors to open source projects appears to be quite skewed. … More than three-quarters of the nearly 13 thousand contributors made only one contribution; only one in twenty-five had more than five contributions. Yet the top decile of contributors accounted for fully 72% of the

> code contributed to the open source projects, and the top two deciles for 81%. This distribution would be even more skewed if those who simply reported errors, or "bugs" were considered: for every individual who contributes code, five will simply report errors. (Lerner and Tirole 2000, 11)
>
> For Apache, the (core) "developers mailing list" is considered the key list of problems to be solved, while other lists play a smaller role. The top 15 developers contribute 83% to 91% of changes. (Lerner and Tirole 2000, 12).

This is not a new phenomenon in the field of open source development. Bill Joy, who was one of the original programmers on the University of California, Berkeley BSD Unix project put it this way,

> Most people are bad programmers. The honest truth is that having a lot of people staring at the code does not find the really nasty bug. The really nasty bugs are found by a couple of really smart people who just kill themselves. Most people looking at the code won't see anything … You can't have thousands of people contributing and achieve a high standard. (Leonard 2000)

Interestingly enough, Bill Joy now works as chief scientist for Sun Microsystems which produces the proprietary, closed Solaris operating system.

While anyone can read the code and contribute to these projects, few people actually do. This is a huge flaw in the theory of open source. There are much fewer eyeballs looking at the code than thought. Furthermore, many of these eyeballs are attached to people who simply lack the skills and talent to contribute. This is much of Bezroukov's argument in his response to Raymond's arguments.

> An important implicit assumption in CatB ["The Cathedral and the Bazaar"] is that open source code is the best thing since sliced bread. It doesn't matter if we are talking about a one hundred line program or a one hundred thousand line program. CatB explicitly assumes that Fetchmail and Linux belong in the same class. In reality, they are quite different projects with very little in common. From the software engineering point of view, there are tremendous differences between a relatively simple project like Fetchmail and a large and complex project like the Linux kernel.
>
> The actual problem is program comprehension. It's always good to have source code but the problem is that often it's not enough. One needs additional documentation to understand a given program. This need grow[s] exponentially with the growth of the code base. (Bezroukov, 1999)

This problem becomes more apparent when looking at the case of Mozilla. Mozilla was an OSS project that had little documentation and few coders working on the project. As documentation and other issues were solved, more development occurred. Code complexity is a problem as it is not merely a matter of the number of lines of code, but rather how each line of code works with each other line. This is where the exponential need for documentation arises as the complexity of a program grows with each additional line.

There are also inherent economic inefficiencies in a product that is given away freely. While there is increasing corporate support for OSS projects, the basic economic incentives to create these properties do not exist under the model of GPL protected software. When sale value cannot be harnessed from software development, a large incentive for production may in fact be missing.

This lack of sale value incentive may contribute to the lack of viable graphical user interfaces (GUIs) available through Open Source. It also may derive from the simple fact that the users most inclined to require GUIs may be those least able to develop software themselves. Consequently, GUIs in products like Microsoft Windows and Mac OS are of higher quality and have greater ease of use than the GUIs available in OSS products. Microsoft and Apple, as private firms which can earn positive profits from sale value, can better aggregate demand for GUIs than can Open Source. The argument is essentially one for strong intellectual property protection. Without the monopoly incentive granted, these firms would not produce these programs because of the large necessary outlay in costs necessary for development.

Many Linux promoters believe that any OSS attempt at creating a GUI at the Windows/Mac OS level is doomed to fail. "Don't they know? The war is over. Microsoft has won" (Mitchell 2001). Because firms cannot reap monopoly profits under Open Source conditions, there are likely losses in efficiency. This also means that Open Source methods may not be appropriate for all forms of software development.

There also exists the problem of free riding, and most literature on the subject downplays the impact, but nonetheless it exists.

> Open source suffers from free-riding in two different ways. First, if development costs exceed individual consumption values, open source developers cannot advantageously provide software, while commercial providers can profitably aggregate demand from many consumers. Second, even when costs are not high, strategic waiting may reduce the probability of provision under open source. (Bessen 2001, 13)

In essence, a common assumption is that the benefit of having an immediate solution is such that it outweighs the cost of development. This is not always true, and as a result Open Source cannot provide solutions to all problems. Also, firms may try and strategize around the assumption that someone else will come across similar problems in the use of the software and that the other firm will fix it instead. These inefficiencies are too often downplayed by the promoters of Open Source, but recognition of such inefficiencies is important in order to understand this development process.

V. Case Study One: Apache

Apache is a particularly interesting case in Open Source. Apache was developed in what was a real grassroots fashion with people who ran web servers adding patches periodically and Apache really developed into a well working, extremely popular product. With an amazing success rate, Apache is perhaps the best example of Open Source Software Development in action. It has a wide base of corporate support, including a massive amount of support from IBM. In order to determine what made Apache so successful, I will look at the history of its closest competitors, its own history, and also the structure of the Apache Software Foundation which oversees its development.

It is important to note, the term "Apache" can be used in several different ways in this form of discussion. Apache is the organization that oversees the Apache web server and there are other parts of the Apache project other than the web server itself. The Apache Software Foundation describes itself as an organization of developers into projects which themselves often have subprojects. (Apache 2003) For purposes of this study, "Apache" will denote the web server project. When referring to the other, related projects of the Apache Software Foundation, I will refer to them explicitly.

Netscape Server History

Netscape is most often associated with its browser product although their web browser was not their primary source of income. The web browser product was to compliment the web server software that they were selling. The thinking was that if they dominated the market for client software, they would also be able to dominate the market for the server software. By embracing and extending the existing open protocols on the

Internet, Netscape hoped to add their proprietary extensions and dominate both the web server and client marketplaces for their profit. Netscape, the company, as well as its client and server products, have had a tumultuous history. I will not rehash the whole rise and fall of Netscape, but its server product's history ties in directly with that of the rest of the company.

The Netscape server was the company's real source of revenue. Initially the web browser software was given away to some groups and sold at around $50 to others. Later, Netscape gave away its browser completely and even released a version of Netscape Communicator as an Open Source project called Mozilla. Its servers, however, were always sold to firms at a cost of over a thousand dollars per license. The latest version, sold by Sun Microsystems as SunONE, has a base cost of $1,495.00 for one license.

Shortly after Microsoft was crowned the winner of the browser wars, Netscape opened the source code to Netscape Communicator, the latest version of its web browsing software. Netscape soon found itself acquired by AOL in 1999. What AOL was hoping to do with Netscape was an open question. AOL had recently made a deal with Microsoft to make Internet Explorer its web browser of choice. AOL did make a strategic relationship with Sun to offer combined business solutions with AOL, Netscape, and Sun Microsystems. The alliance was to produce enterprise computing solutions for the Internet. They were called iPlanet.

> The software that comprises iPlanet has had a tortuous history. Once upon a time there was Kiva, one of the first application servers, which Netscape acquired in stock deal four years ago. When AOL bought Netscape it threw the Kiva server - along with Netscape's other server software - into a joint venture with Sun. The venture got plenty of investment, but never the full autonomy of an independent company. (Orlowski 2001)

The iPlanet alliance eventually fell through as AOL, which became AOL Time Warner, did not want to focus on the enterprise computing market. It sold the rest of the venture to Sun. Sun then branded to iPlanet software to SunONE. Netscape was now gone in all but name. Its proprietary web server is now part of Sun's web application packages.

Microsoft Server History

Microsoft used a tried and true tactic of bundling in order to try and take over the market for web server software. Microsoft sold its server line of operating system products and bundled its Internet Information Server (IIS) along with it. The idea was that for people who ran web servers from computers running Microsoft Windows, they would use Microsoft's free IIS. It was free, it was there, and it was of decent quality. (O'Reilly 1999) IIS was a loss leader for Microsoft used in order to push its server line of operating systems.

It is important to note that Microsoft does not offer just one version of its operating system software. Microsoft offers many versions of its operating system software. There are two versions of software currently being for workstation computers and there is also the server family of products. Currently, Microsoft Windows Server 2003 is scheduled to be sold for a price of $999. (Microsoft 2003) It would be more costly to purchase the Windows server than to purchase a regular Windows machine and put on a free web server such as Apache. There is some question as to whether or not running a windows server is even necessary or cost effective. Tim O'Reilly wrote that the forms of Windows available for the server and for the desktop in the NT line are essentially the same. The NT line of Windows would include Microsoft Windows NT,

Windows 2000, Windows XP, and the upcoming Windows Server 2003. All of these products are based on the same basic kernel and have many of the same features. O'Reilly even claims that Microsoft deliberately crippled certain parts of the operating system in order to force consumers to purchase the server software if they wanted to do things like serving web pages. The cost difference to firms looking for an operating system between a Windows workstation and server products is several hundred dollars. From this we can extrapolate that if the functionality of a large scale server is needed, then the value of the server software to the consumer is actually far greater.

O'Reilly's article in Salon.com is now several years old. In that time it appears that this crippling is no longer built into the current generation of Windows XP workstation operating systems. (O'Reilly 2000) Microsoft Windows XP does ship with IIS. This would lead one to believe that much of the functionality of the server software is built in on a workstation machine. However, it is not clear if an unaltered Windows XP machine with IIS could handle the potentially heavy loads of a web server.

Even if IIS could be had on a machine for no cost above that of a Windows installation, there are still issues that plague it. First off is that there is a cost. Windows, in any incarnation, is not free in either sense of the word. There is the monetary cost. While Windows costs less than the other major commercial web server software available, Windows server software still starts in the $1000 range. There are also questions of scalability and security. Microsoft products have a reputation amongst systems administrators as being unreliable and insecure. A Linux and Apache server would be all free, Open Source software and can give greater scalability and can lower costs for a business.

Development of the Apache HTTP Server

One of the first web servers developed was the NCSA HTTP Server developed alongside NCSA Mosaic at the National Center for Supercomputing Applications (NCSA) at the University of Illinois. The server was released under a public domain license that allowed users to do whatever they liked with the code. The NSCA server immediately became the most popular web server. Rob McCool, the developer of the NSCA server left the project in 1994 and the server languished.

Soon a number of NCSA web server users started adding patches to the server and passing them around as there was no central head of the project any longer. Soon a number of these people decided to create their own distribution of the server with the patches they developed. They decided to call it Apache as it was "a patchy server" and also out of respect for the Apache Indian Nation which has a reputation of skillful warfare and inexhaustible endurance. Less than a year after the group was founded, Apache became the most popular web server. (Apache 2003) Apache now is run on 67% of all web sites with second place Microsoft's servers running on only 25%. (Netcraft 2003)

Development of Apache

Early Development of "A Patchy" Server

> Good programmers know what to write. Great ones know what to rewrite (and reuse). … An important trait of the great ones is constructive laziness. They know that you get an A not for effort but for results, and that it's almost always easier to start from a good partial solution than from nothing at all.
>
> Linus Torvalds, for example, didn't actually try to write Linux from scratch. Instead, he started by reusing code and ideas from Minix, a tiny Unix-like operating system for PC clones. Eventually all the Minix code went away or was completely rewritten—but while it was there, it provided scaffolding for the infant that would eventually become Linux. (Raymond 2001, 24)

Apache did not come forth magically from the ether. When Rob McCool stopped working on the NCSA HTTP Server, the server was still the most used one on the Internet. Many of these users kept developing patches for the server, but no longer had the central leader of McCool and the NCSA to send fixes. Now, a central group started releasing their own patches.

> Using NCSA httpd 1.3 as a base, we added all of the published bug fixes and worthwhile enhancements we could find, tested the result on our own servers, and made the first official public release (0.6.2) of the Apache server in April 1995. … The early Apache server was a big hit, but we all knew that the codebase needed a general overhaul and redesign. … After extensive beta testing, many ports to obscure platforms, a new set of documentation (by David Robinson), and the addition of many features in the form of our standard modules, Apache 1.0 was released on December 1, 1995. (Apache 2003)

Much the way that Linus Torvalds used Minix as a way of developing Linux and Eric Raymond used popclient to develop Fetchmail, the Apache Group used the NCSA server to develop Apache. While the original base of code no longer serves as the backing for Apache, it is important as it served as the foundation from which Apache could spring.

This is much in line with the way many Open Source projects develop. Many times there are multiple versions of the same software available concurrently. This is why development of CVS systems have been so important. In Mozilla, there was the original codebase and then there was the development of the Gecko which was not originally conceived when Netscape first opened the code to the public. Apache was based on code which is no longer in use. The current version of Apache, the 2.0 release (now in version 2.0.44), was developed as a replacement to the version 1.3, but was developed as patches and improvements were still being made to the original base. This is a prime example of how these systems work.

Apache 2.0 was largely rewritten from the ground up to be more stable, capable of handling larger numbers of concurrent page viewers, being more extendable with the other Apache projects, and also to become more platform independent. (Apache 2003) Obviously, these are all large tasks and dropping the 1.3 version, which was also constantly being updated to improve security, was not an option. Improving and redoing is simply part of the Open Source method of development, and Apache is one of the prime examples of this.

Choice of License

Apache is currently available under a licensing scheme similar to the BSD license. This means that the code from Apache can be integrated into commercial products and the code does not need to be shared back with the Apache community. Many firms do sell commercial web servers based on Apache. IBM and Covalent sell web server products based on Apache. These firms have also contributed to the Apache project and release much of the code they developed for their web servers back to the community.

This is because of two important factors discussed in greater length in other portions of this thesis. First is the importance of getting developed code put into the main distribution which is an incentive for any company to release code back to an Open Source Software project. Secondly is the idea of strategic competition. As is discussed in the IBM case study, companies like IBM have an incentive to make Apache the best product possible. The better Apache is, the more likely firms will seek out Apache based solutions, such as those IBM offers through its consulting and value added software packages.

Examination of Apache Structure

Core Group, Core Project

The structure of the Apache group differs from that of other Open Source Software projects. Many OSS projects are overseen by a "benevolent dictator" or an individual that has oversight over the project. Linux is probably the best example of this. Linux is overseen by Linus Torvalds. Torvalds personally oversees what gets included in the Linux kernel. Patches to the kernel are submitted through a process that requires patches to go through layers of checks until Torvalds himself allows its inclusion into Linux.

Apache's organization is more of an imperial senate than that of a dictator. Initially, "a core group of eight users gathered all documentation and bug fixes and issued a consolidated patch" (von Hippel and von Krogh 2002). This core group votes on what does and does not get included into the final patches to Apache that are released to the general public under their auspices.

The members of the overall Apache software foundation comprise a much larger number. The membership is comprised of the people who decide on the http server project as well as on that of other projects, what the foundation should focus on, and also what should be done with foundation money. As the foundation has grown from the original server it has expanded into other projects. These include secure sockets layer (SSL) support, which allows for secure connections for sensitive transactions, and the mod_perl project, which allows Perl scripts to be run on Apache servers for added functionality.

Looking at the list of “Current Apache HTTP Server Project Members” at Apache.org, the number of these members is up to forty-five. According to the policy of the server project,

> New members of the Apache Group are added when a frequent contributor is nominated by one member and unanimously approved by the voting members. In most cases, this "new" member has been actively contributing to the group's work for over six months, so it's usually an easy decision. (Apache 2003)

With the increasing popularity and quality, it is no surprise that this group has grown. In fact, six of these forty-five are affiliated with IBM, two with Collab.net (including Apache co-founder Brian Behlendorf), and two more with Covalent Technologies, which distributes software based on Apache. Only a few years ago, IBM was marketing its own proprietary web server, Domino Go, and now comprises the largest single firm represented.

How this central group decides on what to add to the server is through a unique algorithm.

> The Apache Group is a meritocracy -- the more work you have done, the more you are allowed to do. The group founders set the original rules, but they can be changed by vote of the active members. There is a group of people who have logins on our server and access to the CVS [concurrent versions system, used to manage large software projects] repository. Everyone has access to the CVS snapshots. Changes to the code are proposed on the mailing list and usually voted on by active members -- three +1 (yes votes) and no -1 (no votes, or vetoes) are needed to commit a code change during a release cycle; docs are usually committed first and then changed as needed, with conflicts resolved by majority vote.
>
> …
>
> Anyone on the mailing list can vote on a particular issue, but we only count those made by active members or people who are known to be experts on that part of the server. Vetoes must be accompanied by a convincing explanation. (ibid)

The openness of the process, as well as the code, is unusual even by Open Source standards. For instance, the BSD community has been notorious about not accepting patches from outside the immediate community. This policy is credited by some as

partly the reason Linux has become the free Unix implementation of choice. Linux, on the other hand, is open, but has a more direct pyramidal shaped structure requiring patches to go through many layers of management until Torvalds can give final approval, which he alone can do.

Related Projects

Part of Apache's allure as a web server is its scalability and modularity. Users of Apache can customize it to suit their individual needs exactly as they wish. This heterogeneity of user needs is often uniquely suited to OSS forms of development. The Apache Software Foundation has fourteen software projects other than that of the HTTP server project. Many of these projects have subprojects of their own, and there are also various "sister projects" that extend the capabilities of an Apache based web server solution.

These projects range widely. One such project is the Incubator project. The Incubator project:

> Provides an entry path to the Apache Software Foundation for projects and codebases wishing to become part of the Foundation's efforts. Code donations from external organisations and existing external projects wishing to move to Apache will enter through the Incubator. (Apache 2003)

The Incubator is nothing but subprojects. Any new extensions that want to become part of a distribution of Apache are added through this project.

One of the most popular extensions to the Apache server is the mod_perl extension. This allows programmers that use the popular and powerful Perl scripting language to use Perl in the Apache server. This extension allows for Perl script to be used in all aspects of the server setup. It allows for the creation of dynamic web sites, for creation of further custom modules, and even for configuration of the server itself. "Web

sites like Slashdot and Wired Magazine use mod_perl" (Apache 2003). Perl extensions to the Apache server are often used with databases, such as MySQL, in order to create more dynamic pages.

There are two sister projects which both give security support for Apache. These are the mod_ssl project and the Apache-SSL project. These allow for the creation of secure connections from an individual's browser to the server through SSL, which is the secure socket layer used by all major browsers. This gives those running servers a free option for the running of a business such as retail or other businesses that require secure transactions. These projects are all Open Source and can are available at zero cost. Commercial web servers that have security features have costs starting in the $1000 range. It is important to note, that many firms that use Apache are willing to expend the money to have these kinds of business features, but feel that they are getting a better value by using Apache. As such, many commercial firms, such as Covalent and IBM, sell Apache based products with these packages to firms along with value added support and additional software.

These packages can be mixed and matched to create a more satisfactory web server for an end user. The problem with a closed response to heterogeneous user demand is that it is costly to fulfill all the various user demands, especially when price discrimination is not possible.

> Importance to a customer of the difference between a product designed for an average user and what a given firm or individual really wants will of course vary. An "almost-right" basketball shoe may be reasonably acceptable to a weekend athlete but at the same time totally unacceptable to a professional player.

> The partial response to the true heterogeneity of customer need just described has historically made good economic sense. From the manufacturer's point of view, when it is costly to design, produce and advertise products, it may only be

> profitable to identify and serve a few market segments with products responsive to average within-segment needs. However, recent technological advances have reduced the cost of designing and producing products for "markets of one." As a result it is now feasible to better satisfy customers with needs that deviate from the market segment average. (Franke and von Hippel 2002, 3)

Franke and von Hippel go on to discuss the variations in user demand with Apache security software, but the idea can apply more broadly to Apache and to Open Source as a whole. Apache is unique in its ability to appeal to these "markets of one." Its nature as an OSS project and its ease of development through its "complete toolkit for user innovation for Apache users" allows for these users to be reached (ibid, 9).

These various projects add value to the Apache server. Together, these packages are put together with the server to create a valuable and powerful server. Many of the top sites that use Apache as their server also use the ssl and perl extensions in order to create dynamic and secure server solutions. These solutions create great value to users of Apache software and help explain the immense popularity of Apache.

Conclusions

Apache is a prime example of Open Source at work. It arose from the ashes of another program, reusing code rather than reinventing the wheel. It holds a strong majority of web servers and has held up against competition from the likes of Microsoft. Apache is scalable, modular, stable, and free. It serves as a shining example of the amazing capabilities of Open Source Software development.

VI. Case Study Two: Mozilla

Netscape was the first company in the "dot com" boom. The creative minds that brought the Mosaic browser to life went off to start a company designed to harness the power of the World Wide Web. Netscape Communications was a huge hit by nearly any standard. Netscape Navigator, their flagship product, was an immensely popular product with dominant market share. Their successful IPO was the first to take off in the dot com boom. Netscape seemed to be unassailable in their market position and seemed to be an unstoppable force in Internet technology. That is until Microsoft decided to enter into the browser market.

Netscape started losing its market share and decided to take drastic measures. It would open the source code to its flagship then product, Netscape Communicator[1], in the hopes of staving off the likes of Microsoft. Mozilla, as they called their new Open Source suite of Internet software, is a novel experiment in corporate involvement in the Open Source community. In this case study, I will discuss the history of Netscape, asking why they chose to open Mozilla, look at the history and many problems of the Mozilla project, and look at the current state of Mozilla and the attached projects.

History of Netscape

"Netscape was founded in April 1994 as Mosaic Communications Corp. Mosaic was the name of the browser that transformed the Internet from an academic tool into a mass medium" (Yoffe and Kwak 2001, 1). Netscape built its own web browser similar to

[1] Netscape Navigator was the web browser that Netscape sold and made available for download. Netscape added more and more features to the product, like e-mail and newsgroup reading, so that it became Netscape Communicator. At their cores, Navigator and Communicator are web browsers, but Communicator was more than merely the name given to Navigator 4.0. These products are not fully interchangeable, but for purposes of this paper they essentially are.

Mosaic but superior in quality. Netscape Navigator became immensely popular because of its technical superiority over Mosaic and other Mosaic clones, but also because of innovations that added value to the web by use of plug-ins.

> Navigator 2.0 was a breakthrough product in a number of ways. In addition to e-mail and newsgroups, it supported Java, a programming language recently developed at Sun. This allowed small interactive applications known as Java "applets" to be downloaded as part of a Web page. Navigator 2.0 also featured a plug-in application programming interface (API), which enabled third-party applications, such as audio and video players, to work seamlessly within the browser. (ibid, 3).

The value of plug-ins was that web pages were no longer just linked text documents, but could have animation, sound, and more artistic design incorporated into the page. Web pages were no longer to be static textual pages, but were now a completely new medium of communication.

Netscape rapidly gained market share in the quickly growing browser market. Soon Microsoft saw this as a threat to their core business. Netscape Navigator was a cross platform product and worked on Windows, Mac OS, and various forms of Unix. Through its ability to run programs on top of the web browser, such as Java applets and plug-ins, it became a layer of code similar to an operating system itself. Microsoft worried that web browsers could displace an operating system altogether and saw Netscape Navigator as a direct competitor to Microsoft Windows. Microsoft released its first version of Internet Explorer in 1995 and started integrating the browser into its operating system with Microsoft Windows 98. (ibid)

Microsoft's bundling of Internet Explorer into Windows has been an issue of legal contention with the U.S. Department of Justice and the attorneys general of several states. Nonetheless, while the legal battles raged, Microsoft gained significant market share against Netscape. Soon Microsoft was the dominant player in the web browser market.

Netscape, conceding loss on the browser battlefield, decided to take drastic measures. Netscape chose to open the code to Netscape Communicator and started The Mozilla Organization in order to oversee the project.

Why did Netscape Choose to Open Mozilla?

There are many reasons that Netscape chose to open up the code to Netscape Communicator. Some are the basic benefits of Open Source Software Development that I have already described in some detail. These include a more rapid form of innovation, better feedback from users, and reduced costs of development. Netscape, with its rapidly declining market share, had additional to gain from opening the code base. It was staving off an attempt by Microsoft to close up the web standards and to try and take back market share by increasing its marketing efforts.

Both Netscape and Microsoft were continually trying to increase their market share by various means. One such mean is that of locking in the user to a particular platform. Both Netscape and Microsoft added proprietary extensions to their browsers in the hopes that web page writers would write code specifically for one browser or another. These extensions would allow page designers to be more creative with various design elements. These proprietary extensions meant that pages would have to be designed for either Netscape or Internet Explorer, but pages designed using these extensions would not work on both. The idea was that writers would write pages with the proprietary extensions of the dominant browser. Once pages were written in the style of one particular browser or the other, more users would flock to that browser creating a positive feedback loop.

> For Netscape, the issue was less browser-related income (never more than a small fraction of their revenues) than maintaining a safe space for their much more valuable server business. If Microsoft's Internet Explorer achieved market dominance, Microsoft would be able to bend the Web's protocols away from open standards and into proprietary channels that only *Microsoft's* servers would be able to service. (Raymond 2001, 173; italics author's).

If Netscape wanted to maintain its server market against the onslaught of Microsoft, it had to maintain its browser market as well. Netscape believed that if it were to open the code to Communicator, it would be able to stay ahead of Microsoft in terms of product development, but also would be able to regain its market by making the software itself more desirable to customers (Netscape 1998). Freeing up the software should have done this.

> By open-sourcing the still widely popular Netscape browser, Netscape effectively denied Microsoft the possibility of a browser monopoly. They expected that open-source collaboration would accelerate the development and debugging of the browser, and hoped that Microsoft's IE would be reduced to playing catch-up and prevented from working exclusively defining HTML. (Raymond 2001, 135)

How well this actually worked, I will examine further in this study. Raymond, who wrote this in 2001, seems to have underestimated the problems that would plague this project for some time.

> This strategy worked. In November 1998 Netscape actually began to regain business-market share from IE. By the time Netscape was acquired by AOL in early 1999, the competitive advantage of keeping Mozilla in play was sufficiently clear that one of AOL's first public commitments was to continue supporting the Mozilla project, even though it was still in alpha stage. (Raymond 2001, 135)

There are problems with Raymond's analysis of Mozilla. Raymond does not cite any sources for his assertion that Mozilla increased Netscape market share. Furthermore, it is important to note that there were no significant innovations from Mozilla that were incorporated into Netscape's browser at this time. The first significant innovation that was incorporated into Netscape was the Gecko page rendering engine, which did not come along until years later. Even if Raymond is correct, it is hard to say that the

opening of Mozilla code was the impetus for this increased market share. It could have been a result of any number of things, but the opening of Mozilla seemed to be more of a public relations move than a real strike against Microsoft at this point as evidenced in part by the lack of actual support Netscape gave to the project.

A decision this large for a widely followed company like Netscape was huge news. Netscape was the first darling of the dot com era and its battle against Microsoft was a likened to the struggle of David and Goliath. Could this be the moment that David picks up the stone and hurls it at Goliath? Netscape's own press indicates so. In their announcement they quote Linus Torvalds, the creator of Linux, and Eric Raymond, author of *The Cathedral and the Bazaar*, in order to help build more buzz around their new decision. (Netscape 1998)

Publicity is important to any firm in any business. Netscape's decision to take their code for Netscape Communicator Open Source was a bold public relations move. The decision was unforeseen by the press and completely novel. Never before had a successful commercial project simply become open. Netscape was now giving away that which made it famous and was the product that people associated with the company. It was a public relations dream. Netscape chose to call the project Mozilla, presumably from a combination of Mosaic, the first widely used web browser and the original name of Netscape, and Godzilla, the lizard like creature that terrorizes all who stands before it.

> A week or two ago we all sat around and tried to think up a name for the client; we can't call it Mosaic, because that's the name of the company. The marketroids had all kinds of silly suggestions like Cyber this and Power that and blah-blah Ware. Then someone said something about crushing NCSA Mosaic, and I blurted out "*Mozilla!*" Everyone seemed to like that, so I think that might end up being the official name of the browser. (Zawinski 1994; italics author's)

It turned out that the official name of the browser ended up being Navigator, but the nickname of Mozilla stuck. When it was time to name its Open Source project, Mozilla was the natural choice.

History and Problems of the Mozilla Project

Mozilla certainly did not live up to its public relations hype. The project was plagued with problems. The largest of these was that Mozilla simply did not work. Mozilla did not have a working 1.0 version of its software until June 5, 2002 (Mozilla 2003). That is more than four years after the announcement of the creation of The Mozilla Organization. Part of this problem stemmed from that there was a good deal of cross licensed code that Netscape was not at liberty to release publicly.

> Mozilla initially had difficulty attracting the level of outside contributions that was expected. Mitchell Baker, "Chief Lizard Wrangler" of mozilla.org expressed the views that "the public expectations for the Mozilla project were set astoundingly high. The number of volunteers participating in the Mozilla project did not meet those expectations." … After one year, one of the project leaders quit, citing lack of outside interest because of the large size, cumbersome architecture, absence of a working product, and lack of adequate support from Netscape. (Mockus, et. al. 2000)

Jamie Zawinski, one of the first employees hired at Netscape, was the project leader that resigned a little over a year after the announcement. In his resignation letter, he lists five of what he calls excuses for the failure of Mozilla. These excuses run from the size of the project, to the lack of working code, to the failure of Netscape to commit to the project. Zawinski paints a picture of a company that grew to be a mess and stopped being the innovative startup company it once was.

> What we [Netscape] did from 1996 through 1999 was coast along, riding the wave caused by what we did before.
>
> Why? Because the company stopped innovating. The company got big, and big companies just aren't creative. There exist counterexamples to this, but in general, great things are accomplished by small groups of people who are driven, who

> have unity of purpose. The more people involved, the slower and stupider their union is. (Zawinski 1999).

But there were problems that Zawinski did not even mention. One such was a backlash against the initial license that Netscape released the code under. People said that it was not open enough and left Netscape too much leeway in what it could do with the code that was developed through the Open Source community. Later it was replaced with a license that more resembles the LGPL, but there remained a feeling of mistrust between the community and Netscape that lingered for some time.

Mozilla is a huge project. "Apache had about … 220,000 lines of code… In contrast, Mozilla consists of 78 modules … some of which are much larger than the entire Apache project" (Mockus, et. al. 2000). This code had not grown organically in the Open Source community as Apache had. Netscape had gone through four complete versions of its web browsing software before releasing this mountain of code to the public. Few outside developers were willing to sort through the code to work on a product that did not even work itself. At the time of Zawinski's resignation, "the contributors to the Mozilla project included about a hundred full-time Netscape developers, and about thirty part-time outsiders" (Zawinski 1999).

Part of this problem came from the lack of proper documentation and tools. Code this complex needed to be well documented and divided into smaller groups so that sense could be made of it. It is tempting to think of Mozilla as one large program, but it really descends from Netscape Communicator. Communicator was more than a web browser. Communicator was a web browser, e-mail client, newsgroup reader, web page authorship program, and so on. Netscape could not merely put the code online and hope for people to sort through the code without proper guidance. Mozilla was a mess of undocumented

code, and those in charge of the project felt that changes had to be made, and these changes seemed to have little to do with what Netscape was planning for Netscape Communicator 5.0.

Netscape did not seem to be embracing the project either. Netscape kept developing their same line of products and was working on Netscape Communicator 5.0 which was based on a completely different code base than was Mozilla. Later, Netscape scrapped 5.0 in favor of a Netscape 6.0 based on Mozilla and the Gecko page rendering engine developed for Mozilla by Netscape and the Open Source community.

Mozilla certainly got off to a slow start. It took four years to get to a full 1.0 release of its software. Still, during those four years, Mozilla had undergone a massive transformation. The code was finally split up, documentation had been issued, and a number of Open Source programs were written to assist in development. Mozilla is now one of, if not the, largest Open Source project. It encompasses millions of lines of code, there are thousands of active users reporting and fixing bugs, and most importantly it is now a working program.

There were a number of important steps that The Mozilla Organization took to get to this point. First it redesigned much of the original code to make it run better and to be easier to debug and rework across platforms. Mozilla is designed to run on Windows, Mac, and Unix systems. Cross platform development was facilitated by further modularization of the code base. Some of these major changes included the development of the Gecko web page rendering engine, which is used to load the web pages once on a computer, and the XUL language for designing user interfaces. These made writing code for Mozilla much easier, and submissions increased as a result. (Reis, et. al. 2002) These

improvements were very time intensive and had set back the initial release of Mozilla by several years.

The other major factor in increasing submissions was the writing of ancillary programs which helped push the development along. These programs were written specifically for use with Mozilla development, but have become important parts of many other Open Source projects. Two of the most popular are its own Concurrent Versions System (CVS) and Bugzilla. CVS works as a code repository for users and developers to download and test code, but also to update the code themselves. Bugzilla is the bug reporting tool that was developed for Mozilla in order to properly get bug reports to the people best qualified to fix them. It allows individuals to take responsibility for fixing certain bugs as well as allows bug posters to check on the status of the fixes. Bugzilla also has a search mechanism so that redundant bugs do not get posted. It is a very effective and open system for debugging software. CVS and Bugzilla, combined with the other tools that The Mozilla Organization has developed for development, are potent tools in managing a large Open Source project.

Current State of Mozilla

Mozilla is now a fully featured system and can finally be called a success. After a rocky start and four years of relative stasis, Mozilla is back with a vengeance. Mozilla is considered by many to be a better web browser than Internet Explorer and has features that Internet Explorer does not. Many Windows users are switching to Mozilla from Internet Explorer because of features such as faster page rendering, built in pop-up blocking, tabbed browsing, better printing, and easy text resizing. “For most of my

browsing, I've moved from IE to Mozilla. The way I see it, Microsoft has simply dropped the ball, and there's no reason I should suffer for it." (Dvorak 2002).

In true Open Source fashion, other projects have sprung up around the Mozilla project. Phoenix, recently renamed Mozilla Firebird, is based on Mozilla and is designed to be just a small, fast web browser. It has many of the features of Mozilla in terms of browsing ability, like the speed gain over IE and pop-up blocking, but is free of the mail, newsgroups, and page authoring programs. It is gaining popularity as a simple replacement for Internet Explorer. There are other web browsers based on the Gecko web browsing engine including Galeon which was written specifically for Linux.

Finally, Netscape is now using Mozilla code in its own products. The newest browsing software from Netscape is based on Mozilla code, and has gotten excellent reviews. It has been a long strange trip for Netscape. From a small, innovative startup called Mosaic Communications, to a successful IPO, to battling Microsoft, Netscape is now a division of AOL Time Warner. Now users are slowly moving back to Mozilla from Internet Explorer and only time will tell what will happen next. The failures and successes of Mozilla make for an interesting study of Open Source in action. I think it is best summed up by the conclusion of Jamie Zawinski's resignation letter:

> Let me assure you that whatever problems the Mozilla project is having are not because open source doesn't work. Open source does work, but it is most definitely not a panacea. If there's a cautionary tale here, it is that you can't take a dying project, sprinkle it with the magic pixie dust of ``open source,'' and have everything magically work out. Software is hard. The issues aren't that simple. (1999)

Open Source is a powerful tool for software development, but the case of Mozilla shows that it is not as simple as releasing some code and hoping for development to just happen on its own. The issues certainly are not that simple.

VII. Case Study Three: IBM

One of the biggest supporters of the Linux and Apache projects has been IBM. Why IBM, a company traditionally thought of as a stuffy corporate Leviathan, has taken to a form of development which is traditionally associated with lone, independent hackers comes as a mystery to many, and I hope to unravel this mystery through examining this phenomenon. This can be determined by looking at the public relations boost that IBM has received from this move, the benefits in cost reduction for IBM, and the industrial structure effects of an Open Source embrace that benefit IBM. After examination of the benefits of Open Source to IBM, I ask and attempt to answer why Open Source was not the modus operandi from the very beginning. Through this examination, I will show that IBM exhibited the use of strategic behavior in order to gain market share in the server industry.

Public Relations Benefits

Pundits had thought that the market for mainframes was dead because personal computers gave computing power to individuals. Mainframes are powerful computers that sit in the back offices. Front office workers would sit at "dumb" terminals that took advantage of the computation power of the mainframe. With personal computers increasing in computing power and decreasing in cost, the need for mainframes declined. There was no need for a powerful computer in the back that did computation when there was one on a user's desk. That thinking pervaded corporate computing strategy until the Internet gave rise to a new way of organizing computer systems. Now old mainframes have been reborn as servers and now service web sites, large databases, and streaming

media to personal computers. Firms demand server reliability for these applications because time the server is not working means lost sales, angry customers, and lost productivity for the firm. (Leonard 2000) IBM, once the biggest player in the mainframe market, also needed to be reborn in the server market, and they needed to send the message that the IBM of today is "not your father's Big Blue." IBM is saying that they not only "get it" in today's market, but they get it well enough that they feel confident in giving away some of their most valuable assets.

Perhaps IBM's greatest strength was that it sold a customer the total package. From the hardware to the operating system to the applications that ran on a server to the person who came to your office should anything go wrong, customers knew that IBM stood by its products. IBM's new strategy now eliminates the operating system and the middleware from the equation. Software such as the operating system and other necessary software like web serving is no longer a field of competition. The operating system is no longer a proprietary IBM developed operating system, such as AIX or OS/390. It is now Linux. The middleware that runs the web services is not IBM's proprietary Lotus Domino Go, but the Open Source Apache Web Server. The message this sends to a firm considering an IBM package is that IBM is willing to forgo profits that could be earned by selling additional software. IBM does this because it believes that it will make up for this unearned revenue because, by selling a better system at a better price, firms will continue to bring their business to IBM. IBM has invested over $1 billion and will invest over $300 million more into developing software from which it will never see a dime in sale value. It believes that the services and hardware that it will sell alongside the software will greatly outweigh the costs. (IBM 2002; Leonard 2000).

It also gives IBM a new selling point in that it now offers an open standard with greater network-effect benefits to the consumer. IBM itself benefits from other Open Source related network-effects, which will be discussed further in this paper. By presenting a standard that does not lock in users to any particular software company, unlike proprietary Unix systems like Sun Solaris or HP/UX or Microsoft Windows NT/2000/Server 2003, IBM dramatically cuts costs for firms. Because Linux and Apache are free, there are no built-in obsolescence costs for the firm to bear. Firms can feel comfortable in that they are given software that will not be made obsolete in a year just so IBM can sell them an upgrade. IBM receives no revenue from the sales of these products and thus has no incentive to purposefully make software that is rendered obsolete in a year. Firms considering the various offerings in the market are aware of this. The idea that other operating systems and platforms have been built with planned obsolescence in mind is not a new one. Many accuse Microsoft of purposely pushing substandard software onto the market in the hope that they can make money by later releasing upgrades that provide the usability and quality that should have been in the original, and then charging for them (Slashdot 2001). Conversely, Linux, lacking the profit motive, does not have the planned obsolescence issues.

One of the greatest benefits IBM receives is the intangible benefit of goodwill on the part of the rapidly growing Open Source community, which may come back in sales for services. Raymond calls this the "Give Away the Recipe, Open a Restaurant" model of business (2001, 136). "The market-building effect of open source can be extremely powerful, especially for companies that are inevitably in a service position to begin with" (ibid, 137). The giving away of the software in order to gain a larger market for services

is a business model that Red Hat currently follows and is also used by other firms which may contribute to Open Source projects. These firms feel that the services they provide become more valuable as a result, and that there is a benefit to gaining interest in their services by providing the software freely—the way that Red Hat gives away distributions of the Linux operating system in order to drive business for their services.

Benefits of Open Source

IBM benefited directly through its Open Source support. These benefits arise from network effects, long term cost cutting, independence from another firm's proprietary standard, and the ability to take advantage of preexisting software to enhance the value of their hardware.

Network effects are positive externalities that arise from widespread use of a product. For instance, the more people that use e-mail, the more valuable e-mail becomes as a tool. This is because the more people that use e-mail, the more people with whom one could communicate. While this is not unique to the world of computing, its effects are most evident in the information technology field. In user-developer situations, the more people that use a particular piece of software, the more development occurs, the better the product becomes, and the more people then use it (Raymond 2001; Johnson 2001). This cycle is fundamental to the theories of Open Source. The more IBM supports Linux, the more people use it, and the more people use it, the more sense it makes for firms to purchase Linux solutions. This benefits IBM as one of the premier providers of Linux solutions, such as their servers, consulting services, and application software that runs atop Linux.

Open Source also allows IBM to cut costs. IBM now has thousands more programmers from across the world working for free to make Linux a more stable, better product. Plus, there is already a good amount of software available free for Linux and Apache. IBM's move to these platforms instantly gives IBM a library of software to offer its clients at no additional cost to IBM. This software now no longer has to be produced in house, which means that this frees up money which IBM can to put into other areas of research and development. They are no longer doing redundant work developing software that already exists. It is also important to note that much of this software developed is very relevant to customers of servers, the very market that IBM is trying to capture. Linux is often used as a hosting platform for Internet applications such as Apache and many scripting languages such as PHP and Perl. These are all preexisting programs and platforms that are still constantly being improved upon. These programs, as well as countless more, are now available to IBM. (Raymond 2001)

When IBM was considering Linux, it had three alternatives to using Linux as the operating system. First, they could have used another free distribution of a Unix variant, like the University of California, Berkeley Unix, better known as BSD for Berkeley Software Distribution. However, IBM would not have wanted to use BSD because Linux is easier to port from one hardware platform to another than BSD generally is and BSD does not have the same name recognition that Linux has.

The other two options were to continue to develop their own operating system or to purchase a proprietary platform from another firm. The analysis given in regards to cost cutting plainly makes development of their own proprietary operating system a less viable candidate than Open Source because of the inherent redundancies in reinventing

something that is available for free, but the use of another proprietary platform has pitfalls of its own as well. While it could potentially have been cheaper to use a third-party product such as another form of Unix or Microsoft Windows, IBM would then be locking itself into a proprietary mess of its own. This could have potentially lowered costs as it defrays onto another firm development costs. The other firm could then further defer its costs over a larger customer base. It would however be putting its future at the whims of a monopolist who controls the operating system for any particular platform. Also, IBM has had its own problems in developing software with Microsoft. Their failed OS/2 platform was originally meant as a joint venture between the two companies when Microsoft later dropped the project in favor of developing Windows (Wendt 2003).

Server Industry Structural Issues

IBM's push into Open Source seems to have had an effect on the entire market for servers. While the industry is complex and sells a number of different products, a relatively small number of large firms dominate the industry. These are IBM, Sun, HP, and Dell. IBM, Sun, and HP have their own proprietary Unix systems, but they have all been making inroads into the world of Open Source. (HP 2003; Sun 2002) I believe that IBM's integration of Linux into its product and services line has fundamentally altered the nature of the entire server industry.

There are now two major types of server operating systems available. One is Microsoft Windows and the other is Unix systems, including Linux. Some software, like Apache, will run on both platforms. Lately, both Windows and Linux have been gaining market share at the expense of proprietary Unix systems like Sun Microsystems' Solaris

and HP's HP-UX (Shankland 2002a; Shankland 2002b). As users migrate towards Windows and Linux systems, Sun and HP are left with decreasing market share for their proprietary systems. Linux seems to be emerging as the dominant player within the Unix field.

The market for servers has become more of a market for business computing services rather than a merely a market for computer systems. Now the package that firms expect includes consulting services of all kinds rather than just services for their back office systems. IBM seems to be leading this trend with its purchase of PwC Consulting from the international accounting firm, PricewaterhouseCoopers LLP (Rohde 2002). This merger it is a strategic move made in order to alter the structure of the market in order to put IBM at an advantage. IBM is moving more into the services industry as they push for a model of what IBM calls "e-business on demand" (IBM 2002). The idea is that more of information technology services will be outsourced and the company to best take care of this outsourcing would be IBM through their knowledge of consulting, technology, and open standards and platforms.

Another way IBM is using Linux to position itself strategically is through the use of Linux as a cost-cutting measure. Earlier I mentioned how Linux is being used by IBM to lower its costs. No longer does IBM have to put as much money into development of its own systems as it once did. Now, with Linux, the costs of research and development into operating systems and middleware drop dramatically. By lowering costs, IBM can lower prices for their systems undercutting the competition. This allows IBM to increase market share at the expense of its competitors. IBM is now in a unique position to offer a la carte value added services such as their WebSphere suite of applications and their DB2

database software (IBM 2002). These applications are examples of proprietary software that run atop Linux and Apache. This differs from its older model insofar as these applications do not compete directly with viable Open Source alternatives. They are instead applications which add value to these existing platforms. Eric Raymond, the Open Source evangelist puts it as such:

> For software and systems vendors competing against Microsoft and its IIS web server, the Apache project is also a competitive weapon. It would be difficult, perhaps impossible, for any other single web server vendor to completely offset the advantages of Microsoft's huge war chest and desktop-monopoly market power. But Apache enables each corporate participant in the project to offer a web server that is both technically superior to IIS and reassures customers with a majority market share—at far lower cost. This improves the market position and cost of production for value-added electronic-commerce products (like IBM's WebSphere). (Raymond 2001, 149)

The effect of IBM embracing Linux can be seen through the reactions of their primary competitors. In recent months, Sun and HP have also started to move towards providing Linux solutions. In February 2002, Sun Microsystems announced that "it will 'aggressively participate in the Linux community,' offering key components of its Solaris operating system for free" (Kary and Shankland 2002). Since this announcement, Sun has made several other announcements about product lines that they are introducing using the Linux operating system. This would include a "purple box" line of Linux-based personal computers, since Sun is often associated with the color purple similar to the way IBM is with blue. (Shankland 2002c) HP has entered the fray with a line of blade servers: slim, modular servers that can be stacked in order to place more computing power in less space (HP 2002). Sun is now selling their own blade servers with Linux as an option (Sun 2002).

It seems that the effect of IBM's move towards an open, less costly system has put Sun and HP in a defensive position. Their proprietary systems seem less appealing

when placed against an open system provided by IBM. No longer can these firms reap profits on their differentiated operating system offers, but now have to compete in price, service, and hardware capabilities alone. IBM must feel that they have an advantage in these fields and garners further advantage by being the first to enter the market. Of these three firms, IBM is commonly seen in the Open Source community as being its biggest supporter and many potential customers probably see IBM as the provider to go to for their Linux server solutions.

Conclusions

In conclusion, in order to maximize its profits and market share, IBM strategically positioned itself in the Open Source community through their support of the Linux and Apache platforms. This gave IBM cost benefits, but also allowed them to restructure the industry in their favor focusing on services and hardware while taking the operating system and middleware completely out of the equation. IBM's unique position in the industry now has far reaching consequences that have yet to fully seen, but the next few years should provide more information, from which we can draw better conclusions.

VIII. Case Study Four: Microsoft Shared Source

Microsoft Corporation is one of the most revered and despised entities in the software world. Since its inception, Microsoft has been a staunch supporter of strong intellectual property rights. Infamously, Bill Gates had even gone to the Homebrew Computer Club at Stanford University to inform them that they were stealing software when they shared Microsoft's first product, a BASIC interpreter for the Altair. Microsoft's stances have changed little since that time. Still, Microsoft is not the polar opposite of the Open Source Software community. Microsoft has begun to share more and more of its source code with academics and its business partners. It even has gone as far as releasing a program under the General Public License (GPL). In this study, I examine Microsoft's enigmatic flirtation with Open Source.

Microsoft's Early Reactions to Open Source

Open Source was immediately seen as a threat to Microsoft's business model. The "Halloween Documents" written by Microsoft personnel starting in 1998 are testament to this. (Open Source Initiative 2003) These documents reveal the way that Microsoft felt about Open Source and how they continue to feel about Open Source Software and Open Source development.

Since the first Halloween document in the fall of 1998 there have been seven more of these so called Halloween documents from Microsoft relating to Open Source. The latest of these documents was a leaked memo dated November 2002. Microsoft denies that they represent the official position of the firm, but many believe that the opinions expressed within are more than just the writings of random technical writers

within Microsoft. These documents give unique insight into the inner workings and thoughts of the employees at Microsoft.

> Open Source Software (OSS) is a development process which promotes rapid creation and deployment of incremental features and bug fixes in an existing code / knowledge base. In recent years, corresponding to the growth of Internet, OSS projects have acquired the depth & complexity traditionally associated with commercial projects such as Operating Systems and mission critical servers.
>
> Consequently, OSS poses a direct, short-term revenue and platform threat to Microsoft -- particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat. (Open Source Initiative 2003, Halloween I)

This is the executive summary of the first Halloween document of Microsoft's. It is telling that Microsoft seems to have an understanding of the benefits of Open Source to developers and how it sees Open Source as not just a short term fad, but as a "long term developer mindshare threat" (ibid). Microsoft goes so far as to admit that OSS projects have achieved commercial quality and thus pose a legitimate threat to their business.

Microsoft also admits "Loosely applied to the vernacular of the software industry, a product/process is long-term credible if FUD tactics can not be used to combat it. … OSS is Long-Term Credible" (ibid). FUD stands for Fear, Uncertainty, and Doubt. FUD tactics are considered typical Microsoft marketing tactics. Whenever a new product is introduced into the market, there are understandable trepidations about it. Entrenched firms in a particular market try to place public relations attacks against the incoming firms and products. Microsoft notoriously used FUD to try and scare customers away from new products and into the "long-term credible" arms of Microsoft. Microsoft still attacks OSS using FUD, but does so by attacking the GPL directly as a license. By

referring to the GPL as “viral” they hope to scare potential customers away from using software licensed under the GPL.

Microsoft discusses certain issues with specific software too. Halloween II is largely about Linux. Linux has achieved a certain celebrity amongst OSS projects. It has gained market share against other forms of Unix and even against Microsoft Windows. Its executive summary in the opening of the memo says,

> The Linux OS is the highest visibility product of the Open Source Software (OSS) process. Linux represents a best-of-breed UNIX, that is trusted in mission critical applications, and - due to it's open source code - has a long term credibility which exceeds many other competitive OS's.
>
> Linux poses a significant near-term revenue threat to Windows NT Server in the commodity file, print and network services businesses. Linux's emphasis on serving the hacker and UNIX community alleviates the near-medium term potential for damage to the Windows client desktop. (Open Source Initiative 2003, Halloween II)

This is interesting in two ways. First it acknowledges the threat of Linux not only to Microsoft but to the overall Unix market. As I pointed out in my case study on IBM and their embrace of Open Source, commercial forms of Unix have become less and less viable. As a result, IBM has shifted its focus from its proprietary AIX system to Linux. Interestingly enough, this document seems to have predicted the IBM move to Linux. In looking at Linux business models, it acknowledged that, “IBM is most capable at capturing revenues from all 4 of the business models associated with Linux.” Also of note is that Microsoft does not see Linux as a near term or medium term threat to the desktop market. They do admit that Linux is a something of an alternative to Windows on a desktop environment, but they do not think that windowing systems like KDE and GNOME and office suites such as OpenOffice.org make a truly credible threat to Microsoft’s desktop hegemony. (ibid) It does seem that in cases where users may not be

the same people involved in developing, for instance in software such as GUIs or office productivity software, that private concerns such as Microsoft may have advantages in their ability to aggregate demand.

The Microsoft Shared Source Initiative

Microsoft has not backed down from it position as a firm producing closed source products for sale. Software is Microsoft's primary business, and Open Source, which would mean the software would be available at no cost, would take away Microsoft's primary revenue stream. Open Source is simply not a viable option for Microsoft to embrace. Furthermore, Microsoft does recognize that the Open Source model for competing software solutions is a viable threat to its revenue stream. There needed to be some sort of solution. Microsoft came up with its cleverly titled "Shared Source" scheme.

Shared Source, as a concept, is designed to straddle the divide between closed and open projects. The idea behind this is that Microsoft is trying to capture many of the benefits of Open Source, like better user feedback, improved third-party software and drivers, and even greater loyalty on the part of users and developers. In this section I will examine how Microsoft has set up the Shared Source campaign by looking at the licenses used and their restrictions on users as well as Microsoft's outreach to the developer community.

Microsoft has made more and more of its products available with source to varying degrees. Some of these products are available with the restriction that they can be used for non-commercial use in just about any way a user could want. Others require

extensive licensing including the signing of non-disclosure agreements (NDAs). Here is Microsoft's official statement on their Shared Source:

> Microsoft is sharing source code with customers, partners, governments, and competitors. Source code access improves software transparency, raising levels of trust and increasing flexibility. The Shared Source Initiative also preserves software intellectual property rights that have sustained innovation throughout the software industry for more than a quarter-century. (Microsoft 2003a)

Microsoft is essentially trying to have its cake and eat it too. Strong intellectual property rights are essential to the sale value model of business to which Microsoft adheres. Even still, Microsoft recognizes the threat that OSS poses and it recognizes the allure that source code availability presents to the developer community. The Shared Source initiative is an attempt to increase the transparency in its dealings with these developers while keeping its crown jewels under lock and key through copyright protection.

Microsoft Windows CE, the version of Windows for embedded systems such as set top boxes and handhelds is available for academic use with little other restriction. Microsoft even allows redistribution of modified source code and binaries (compiled runnable code) as long as it is then also for academic, noncommercial use. This is something of a radical change for Microsoft. Microsoft Windows CE is based on the same code as the rest of the Windows family.

Microsoft also has opened up to using its code in academic works. The Windows CE Shared Source Academic Curriculum Licensing Program is a program that "gives professors, researchers, and graduate students the ability to create curriculum, including textbooks and other teaching materials, that includes Windows CE .NET source code" (Microsoft 2003a). This is an interesting move as well. Currently, many computer science textbooks on operating systems are based on Unix systems. Linux being a popular operating system for servers is often the operating system of choice for computer

science labs at universities, including the "Berry Patch" at Brandeis University. Because Linux code is open source, universities' curricula are using Linux increasingly. Microsoft, I posit, is using this as a way of gaining mindshare in the academic community through educational materials. Many people, both students and faculty, in the computer sciences use Microsoft Windows as well as Unix variants, including Linux, and often times texts refer to both in explaining various paradigms. By allowing Windows code to be placed side-by-side with Linux code, Microsoft is hoping that students will become more accustomed to Windows architecture than most students are currently. These students will become the developers and system administrators of the future and Microsoft may believe that in doing this they will capture their loyalties for the future. Microsoft does require that these authors pass through a screening process, however, and reserves the right to reject applicants from using their code.

The major licensor of source code from Microsoft is computer manufacturers (OEMs). Microsoft has to be particularly careful with OEMs as they comprise their largest customer base. Microsoft has even admitted that OEMs, if they could credibly threaten to move to OSS alternatives from Windows systems, could comprise a major threat to their bottom line. "Compaq and Dell merely have to credibly threaten Linux adoption in order to push for lower OEM OS pricing" (Open Source Initiative 2003, Halloween II). In order to cozy up to the OEMs, Microsoft has allowed more of its source code to be available. Their system "provides a mechanism for delivering source code for the most current versions, beta releases, and service packs of Windows 2000, Windows XP, and Windows Server 2003" (Microsoft 2003a). In opening up this code, it allows hardware manufacturers to better integrate their architectures with Windows and

thus get better performance and offer better support to their customers than they could otherwise. This is not as flexible as a true Open Source license as the OEMs cannot alter the code and offer specialized versions of Windows specifically for their customers. Microsoft specifies, "Licensees may read and reference the source code but may not modify it."

In addition to providing source code, Microsoft has a developer network that includes tools for development. These tools are things like the Microsoft Developer Network, which allows developers access to many of Microsoft's software products and Microsoft Visual Studio, which is used to develop the software itself and is used by many outside firms for their own software development, and the Windows Application Programming Interface (API), which is a library of classes that can be used to develop programs for Windows. "Microsoft is also making its code easier to read through documentation and colored syntax, making it easier to understand what does what" (Patrizio 2002). This would qualify as what Franke and von Hippel (2002) would call "innovation toolkits." Compare this to the early days of Mozilla when no such toolkits were available. Microsoft, despite having Windows as a closed system, still provided tools such as these in order to facilitate growth of third-party software. Code availability adds value to these toolkits.

Despite all the protestations against the Free Software Foundation's GNU General Public License (GPL), Microsoft does in fact have software released under that such license.

> Surprisingly, Microsoft actually has at least one product supported under the GNU General Public License. A few years back, Microsoft purchased Interix, a developer of Unix-to-Windows code conversion tools. Microsoft continues to sell and upgrade the Interix suite to help move Unix applications to Windows 2000.

> Some of the tools in the Interix suite were released under the GNU GPL that Microsoft dislikes so much, and Microsoft continues to release the code under GPL, honoring the previous license. (Patrizio 2002)

While the source code is available through the GPL, any derivations of GPL licensed software must be covered under its auspices, and another stipulation is that any software covered by it cannot be sold.

Microsoft has not embraced Open Source as a tool for software development, but it seems to have conceded that the old, closed system is not indefinitely sustainable. Even if not freely, opening the code is a step forward for Microsoft. What this will mean for the future of software is unknown. Microsoft's business model, unlike that of IBM or Red Hat, is based on the sale value of software, not on additional services, support, documentation, or consulting. Open Source as a business paradigm takes out sale value from the equation. Time will tell if the sale value model will remain the dominant model for Microsoft.

My Theory as to How Microsoft Can Combat (or Embrace) OSS Development

A question that could be asked at this point is whether there is a way to allow Microsoft to open its code while still retaining its intellectual property rights and thus its revenue stream. There is not a simple answer to this, but I feel that to an extent the answer is, yes. There are certain strategies that can be used in order to take advantage of the resources offered by the Open Source community while still maintaining sale value of the software.

The first one of these ways would be to take a route similar to the one taken by Apple Computer. Apple has recently rewritten its operating system. Mac OS X has been

rewritten from the ground up using BSD Unix. Apple is under no obligation to open the code to OS X because of the stipulations of the BSD license. Nonetheless, Apple makes the source code to its kernel, Darwin, available to the general public. Darwin is available under the Apple Public Source license. The license has been certified as an Open Source license by the Open Source Initiative, the generally accepted certifying body.

Conceivably Microsoft could do the same thing. Currently, Microsoft has released part of the Windows NT kernel under its Shared Source licenses, but not to the general public. Microsoft could conceivably open its NT kernel and take advantage of the Open Source community's free labor in order to improve upon it. This would not be without problems though. Conceivably certain users would be able to build Windows systems without paying any money to Microsoft. This would be unlikely though not impossible. If Microsoft keeps its GUI closed, as Apple does, software written for Windows could require some parts of the system that can only be had through a commercial distribution of Microsoft Windows. Currently, in order to run native Mac OS X software, a user would need a commercial copy of the operating system. However, many in the Open Source community are already trying to emulate Windows' functionality. The WINE (WINE Is Not an Emulator) project is an attempt to run Windows programs on other systems, including Linux. Purchasing Microsoft Windows would then not be necessary to run Windows third-party software. If Microsoft were to make its kernel more open, it could conceivably fuel these attempts and ultimately hurt Microsoft's bottom line.

Another paradigm Microsoft could use is what I call the book paradigm. Open all of the source code, but make sure that it is known that Microsoft owns it. Much the way

that an individual has rights over the book that is in her hands, the individual would be able to see, edit, and recompile the code. However, the user could not redistribute the code and would only have a previously specified number of licenses. That is to say, if one were to purchase one book, that individual would not be empowered to make copies or to change it slightly and pass it off as a completely new work exempt from the copyright. This does have many of the advantages of Open Source for an end user-developer. The user would know that she would own the product outright. No matter what would happen to Microsoft, the code would be available for bug fixing in the future.

This is not without potential perils to Microsoft. Conceivably, users would retain rights over the modifications made to the code and could distribute that code without explicit permission from Microsoft. Much of the revenue stream of Microsoft is based on the sale of upgrades to current systems. If the fixes could be made at no cost to the firms because of patches available on the Internet or developed in-house, then Microsoft could lose this lucrative portion of the business. It would also be unclear about what liability Microsoft would incur over these user patched systems.

Theoretically, Microsoft could stipulate in the license the terms for redistribution, such as allowing for redistribution within the user's own firm or institution and Microsoft exclusively. Open Source licenses, such as the GPL, stipulate licensing for code developed by users through code covered under the license. This would mean that all greater patches would come from Microsoft and it could still retain the upgrade business. Furthermore, many patches, such as security fixes and upgrades to Internet Explorer, are available for free through Microsoft's Windows Update feature. If firms were able to patch things up at their own expense and if Microsoft were able to distribute these

patches, then Microsoft could end up better off than previously. This would not be unprecedented. Historically, user innovations in the cases of the furnaces and even in high-performance windsurfing would typically be given back to the manufacturer. At that point the manufacturer would integrate these new features into the product and users would then purchase the new product that included their innovations. (von Hippel 2002)

Under the book paradigm, third parties could conceivably sell more modularized and specialized versions of Windows. For instance, Dell could sell a Dell Windows and would have to pay a licensing fee for each copy thereof. This is an extension of the idea that one could purchase a book and then resell that book with highlighting and notes. Users would then become more reliant on Dell for future versions of Windows. Of course, this would give Dell greater power in its market and give it something of a monopoly on the operating systems for Dell computers. Potentially this could be an avenue for Microsoft to engage in price discrimination and could raise various antitrust issues with the United States Department of Justice.

Even with these licenses, the incentives for those outside of Microsoft to develop would be less than the incentives in a true Open Source community. Part of the incentive is that revealing innovations early on would increase the likelihood of the innovation being integrated into the central distribution and thus the innovation would spur further innovation. (Harhoff, et. al. 2000; Raymond 2001) This incentive certainly exists with Shared Source. Developers give up their intellectual property rights to their innovations largely in exchange for others giving up their rights to innovations that derive from that innovation. If the rents from this innovation were to be largely captured by an outside

firm, namely Microsoft, user-developers may be less inclined to use their own resources for development if they give up many of their rights to the innovation.

Conclusions

Microsoft may well be the most dynamic firm out there creating sale value software. From its Windows and Office franchise players to Windows CE, the Xbox gaming system, and its online presence through MSN.com, Microsoft is certainly a force in the industry. Open Source is also a powerful force within the software community and has certainly made itself known to Microsoft. What the future holds for the varying forms of software valuation and development is unknown, but Microsoft's memoranda and actions show that it recognizes that Open Source is here to stay.

IX. Case Study Five: Red Hat, Inc.

Red Hat is one of the most interesting companies in the software market today. Their business model is based on giving away their intellectual property at no cost to consumers. They develop and market this software at great expense, but hope to make profits through other means. In this essay, I will look at Red Hat's business model and whether a business model developed around Open Source can be viable for a firm.

Why Businesses Choose Linux Solutions

Linux has concrete advantages for many firms, and by improving Linux, Red Hat hopes to profit. Linux solutions are often preferred to those available through internal operating system development or purchase of a proprietary system because of several important factors. These include lower cost, the high quality of Linux as an operating system, a non-existent incentive for planned obsolescence, and that Linux is a standard across business.

While Open Source Software is only free if your time is worth nothing, the use of Open Source Software, particularly Linux, can be a great cost saver for firms. This is because while the expertise and support required for the use of Linux is not free, additional licenses for use of the software are. Open Source Software is, by definition, free to use on as many computers as one likes and can be redistributed freely. Compare this to any of the proprietary systems. A license is required for all computers that run software such as Microsoft Windows. In theory, a firm could use Linux on its desktop computers. Doing so would mean that they would not have to pay the several hundred dollars per system "tax" to Microsoft. In practice, few firms use Linux on their desktop

computers. Dell experimented with supporting Linux desktop systems for a short while. In 2001 Dell stopped supporting Linux on desktops and laptops citing poor demand, but Dell continues to support Linux on their line of servers (Spooner 2001). For many workstation systems however, Linux is popular and cheaper than other commercial workstation operating systems.

For firms looking to develop standalone electronic devices, such as set top boxes for televisions, personal digital assistants, wireless phones, and even digital wrist watches, Linux provides a low cost alternative. Firms could either purchase an operating system such as Microsoft Windows CE or the Palm OS, but they would be required to pay a royalty for each device produced. An alternative could be producing their own operating system, as many producers of these devices do. This would require diverting resources away from hardware and application development to create an operating system. It would also have standardization issues in that there are no existing developers to create third-party applications for the system. Linux is would have the advantage of being free and having a preexisting library of software and a large number of existing developers who are capable of creating software for the platform. For this reason, TiVo has chosen to run its systems atop Linux and Sharp uses Linux for its Zaurus PDA.

Linux has a reputation of being a stable and reliable operating system. Linux servers can run for extensive periods of time without rebooting while handling heavy loads. Through its rapid development cycle, Linux has become a viable operating system across hardware platforms and is now trusted to run servers for the Fortune 500. It has even become the operating system of choice for technology companies like IBM. Its reputation for stability has allowed for firms and individuals who use Linux to take

advantage of network externalities. Especially in the cases in which firms are looking for embedded solutions, Linux is a preexisting stable platform. Ensuring stability in an internally developed system may present greater risks.

Planned obsolescence has been an accusation thrown at many software companies, not least of all Microsoft. (Slashdot.org 2003a) One of the advantages to the adoption of Open Source Software for firms is that there is no profit motive tied to the software. If Microsoft were to produce a perfect version of Windows, consumers would never have to upgrade and buy new Windows licenses for computers they already own and operate. By releasing new versions and charging for them, Microsoft can make money from existing customers by producing an inferior product—in theory. Like most conspiracy theories, its allure is that it can never be proven one way or another. The contributors to Linux do not own the intellectual property rights to the software, and the sale value of Linux is essentially zero in this respect. Consumers of Linux and other Open Source Software know this. Consequently, many feel confident that they are not being locked into a standard for which they could be paying large fees for an indeterminable amount of time.

Another advantage for businesses is that Linux is already an established standard in the business. This works as an advantage for both original equipment manufacturers (OEMs) like Dell but also for firms who use Linux as either an enterprise solution or as a workstation solution. This is because, as an established standard, there are a large number of applications that run atop Linux and a large number of developers capable to write software to run atop Linux. For an OEM firm such as Dell or IBM, this means that putting Linux on their servers means that they are selling a product that already has a

large base of software available. These are network effects that work to the advantage of Open Source in general and Linux most specifically. The quantity of user-developers ensures that Linux will remain a viable system for years to come. Consequently the viability attracts new users, developers, and OEMs to the platform. This means that a firm considering operating systems will look to Linux and see the large number of trained developers available. Compare this to a proprietary system like IBM's former flagship operating system of AIX. AIX was used only on large IBM servers, and as a result, few developers developed for this system as there was not a large market, and firms looking to purchase a system would often look elsewhere because there would be fewer programmers trained to program on AIX. Firms want this established standard because it is easier to find trained programmers that can develop for Linux than it is to find for a system like AIX of Sun Solaris. Firms also know that their systems are likely to work better together upon using one system across the board, and an open standard means that they would not be locked into using a single OEM for their hardware needs.

Red Hat Business Model

Red Hat's primary products, its line of operating systems designed around the Linux kernel, are free to download and distribute. Red Hat invests heavily in the development of Linux. They have even hired Alan Cox, who is second only to Linus Torvalds in the Linux development hierarchy. This is despite the fact that they earn no revenue from the direct sale of their main product and instead organize their business model around products and services which attach themselves to this software. These products and services are where Red Hat derives its revenue, and the software itself is a

loss leader. The better Linux becomes, the more desirable using Linux becomes, and the more firms will flock to the services Red Hat provides. In this sense Red Hat is more of a consulting firm rather than a software development firm.

Enterprise Solutions

Open Source Software is "free as in beer and free as in speech." That said many individuals and firms still choose to purchase Open Source Software, such as Red Hat Linux, because they value the services that are packaged with purchase of the software. It would be hard to picture any firm putting its data on a server running an operating system that did not have technical support coverage. In publishing its documentation and in selling direct services such as support and updates, Red Hat is filling in this role.

Red Hat now sells three levels of its software. There is Red Hat Enterprise Linux AS, Red Hat Enterprise Linux ES, and Red Hat Enterprise Linux WS. These presumably stand for advanced server, enterprise server, and workstation respectively. These three products are all composed entirely of freely available software, but the AS product costs $1499 for the standard edition and $2499 for the premium edition (Red Hat 2003b). The advanced server, according to the site, "is the ultimate solution for large departmental and datacenter servers" (ibid). The services and support for a server such as this would be something for which firms would gladly pay.

The ES software, costs $349 for the basic edition and $599 for the standard edition. The difference between these two models is the following:

> Basic Edition provides 90-days of Installation and Configuration support (Monday-Friday 9am-5pm ET), and a 1 year subscription to Red Hat Enterprise Network. It is available via download only.

> Standard Edition provides a full year of Standard support (includes Monday-Friday 9am-9pm phone support with 4 hour response (9am-5pm outside North America) and a one year subscription to Red Hat Enterprise Network. Customers ordering Standard Edition will receive a full boxed-product with CDs and printed documentation. (ibid)

This is all while "Red Hat Enterprise Linux ES provides the same core capabilities as Red Hat Enterprise Linux AS" (ibid). This form of price discrimination allows Red Hat to target its consumers based on the value they place on the importance of these platforms to their businesses.

The final version, WS, is focused on workstations. It costs $179 for the basic edition and $299 for the standard edition. The service differences between the two are virtually identical to the ES differences in levels. The WS comes without many of the software applications designed for larger servers which would more likely run the ES or AS versions of the software. A software product which aims for a similar platform, Microsoft Windows XP Professional, costs $299 for its full version when purchased directly from Microsoft. (Microsoft 2003d) The levels of service and support are not directly comparable, but this does indicate that the services and the product that is purchased from Red Hat is in many ways the equal in value to what Microsoft is selling for many consumers.

The Red Hat Enterprise Network, which is referenced in the software packages, is a service that is used to update the software on the systems which run Red Hat Enterprise Linux. This is more than just a simple updating system, as it allows companies to customize the amount of updating various systems have across their information technology infrastructures. It also allows for grouping of computers, for different administrators to have different levels of access, and for scalability in upgrading a large number of systems. While upgrading single systems may be trivial, maintenance of an

infrastructure can become exponentially more complex as size increases. Red Hat's CFO predicts that "subscriptions will be 70% of revenues, and services will be 30%" (Stone 2003).

Consulting Services

Red Hat also sells consulting services which go beyond the regular services for enterprise systems. They provide solutions for companies looking to migrate from proprietary Unix systems to Linux, for firms looking to build IT infrastructures, for large hardware manufacturers looking to add Linux to their line of servers, and even for companies looking to embed Linux into other hardware products. Red Hat has unique insight into Linux as the developer of one of the most popular distributions of Linux. Red Hat attempts to leverage this insight and expertise as a way of earning revenue through its consulting services.

Linux is compliant with the standards for Unix systems. This means that it can run most of the software applications that competing forms of Unix can run. These proprietary Unix systems run on expensive hardware platforms and software updates can easily cost several thousands of dollars. Linux, as Open Source Software, can drastically cut these costs. I will discuss reasons businesses might chose Linux over alternatives in greater depth in the following section. In order to take advantage of Linux, firms need assistance in converting their older systems to Linux. Red Hat provides consulting services for firms looking to migrate to Linux. The cost savings to firms can often be enormous and some of these savings can be captured by Red Hat. One prominent example listed in the 2002 annual report was Amazon.com, which was able to cut their

technology expenses by "$17 million, or 24% from a year earlier – attributed in large part to its shift to Red Hat Linux."

Building an IT infrastructure can be a daunting task. These infrastructures can include web servers, database servers, printer sharing, file sharing, and individual workstations. Mistakes, in terms of incompatible hardware and software, can be costly for firms. Conversely, the correct decisions can improve productivity tremendously. Red Hat sells consulting services which assist firms in making these decisions. They help firms streamline their IT infrastructures by standardizing on Linux and other open standards which can facilitate growth by not locking in firms on a proprietary standard or a number of incompatible proprietary standards. By producing better and better versions of Linux, they hope to increase the desire for firms to build their systems around Linux. Many more firms are looking to Linux because of the advantages that Linux provides over alternatives and Red Hat hopes to capitalize on this phenomenon.

The industry for large central computers has changed dramatically in the past few years. The cost of computing power falls rapidly according to Moore's Law, and companies are looking to take advantage. Linux runs on cheaper computing architectures, such as the ubiquitous x86 platform, on which the vast majority of personal computers are based. Most firms in the industry for servers had run proprietary forms of Unix on their own, very expensive, hardware systems. Development of these hardware and software lines is sufficiently expensive that there were only a handful of manufacturers.

Companies such as Dell have partnered up with Red Hat in order to develop a line of servers which use cheaper hardware and use Red Hat Linux as the operating system.

This allows companies seeking solutions for servers to use Red Hat's services in order to develop their own lines of servers. Partnering with Red Hat also allows for easier embrace of Linux as a way of undercutting the competition for operating systems in the server arena. Red Hat now currently has relationships with firms like IBM and HP, firms that have been long established firms in the server industry and still produce their own proprietary Unix systems. (various sources)

Computing power has become increasingly pervasive. Even watches and cell phones no longer function in just one way. Very often these devices are computers that run atop embedded operating systems such as Microsoft Windows, Palm OS, or even Linux. For firms that develop hardware solutions, Linux is an attractive alternative to systems on which firms would have to pay royalties for each device made or to developing their own operating system. Firms can hire Red Hat to bring Linux to their device and focus instead on development of applications and of hardware rather than on development of an operating system.

Other Related Products and Services

Red Hat also makes money through services such as certification and education programs and also through the sale of Red Hat branded products. The education and certification program, known as Red Hat Certified Engineer (RHCE) and the Red Hat Certified Technician (RHCT) involves courses taught around the country and certification examinations. These certifications also need to be renewed by individuals who wish to retain the certifications. The RHCE is considered to be one of the "one of the ten hottest certifications in the IT industry," and the certification allows individuals to

stand out in the IT job market (Red Hat 2003a). The prices for the training and examinations start at around $750 and go above $2000.

Red Hat also sells branded merchandise in its online store. Since the only intellectual property on which they claim ownership is their trademarks, Red Hat reaps revenue by selling these products. They sell shirts, baseball caps, mouse pads, and even red fedoras. These promotional items have a certain amount of "geek appeal," and there are people who purchase such items.

Red Hat as a Business

Currently Red Hat is amongst the most popular distributions of Linux. It is impossible to determine market share of a product that can be distributed freely, but it is commonly believed that Red Hat is the dominant distribution of Linux in the United States and for business Linux servers. Red Hat has advantages over other distributions of Linux in many areas that appeal to firms. Its popularity allows for network externalities above that of simple Linux use and as a for-profit institution it is more responsive to its customers than other forms of Linux which may not have the level of commercial support. Nonetheless, there are potential pitfalls that lie ahead for Red Hat. In this section, I will examine how Red Hat has performed financially and look at Red Hat's competition.

Financial Analysis

Red Hat has been successful in recruiting new partners in order to extend its reach across the enterprise computing market. Red Hat has recently made a new strategic

alliance with Oracle and Dell in order to allow the firms to sell what they call enterprise ready Linux solutions. In essence, with the database software from Oracle, the hardware from Dell, and the operating system from Red Hat, firms have ready to use computing solutions for their needs. Red Hat has also made alliances with IBM, HP, and Intel. (Red Hat 2003d) This increasing number of partners leads to an optimistic outlook on the future of Red Hat.

Red Hat's financial picture is a bit different, however. While it did seem to be turning a profit in the period ending November 30, 2002, this was a profit of only $214,000. Its latest financial numbers indicate that Red Hat is back in the red. Red Hat's current earnings per share is -$0.04. It is trading closer to its 52-week high and many analysts think that Red Hat is overvalued. (Yahoo! Finance 2003)

> "The revenue growth isn't particularly impressive," says Paul McEntire, portfolio manager of the Marketocracy Technology Plus Fund (TPFQX), which has owned the stock in the past. Moreover, he says, Red Hat's financial results don't persuade him that it can be solidly profitable in the future. Mostly, he worries that it would take only a little price competition from Microsoft (MSFT), which goes up against Linux in the operating-system market, to see the return of red ink. Notes McEntire: "Microsoft hasn't really responded to the Linux threat yet."
>
> STRATEGIC RELATIONS. Given Red Hat's small profits and moderate revenue growth, many analysts think the stock is expensive. After riding the dot-com roller coaster, it was a relatively comfortable $5.49 at Feb. 20's close. From a low of $3 in late 2001, it has stayed roughly in a range between $4 and $6 for the past two years -- not bad, considering that the tech-heavy Nasdaq has fallen about 40% over that period. But that also doesn't make it especially cheap.
>
> In a Jan. 29 research report, Goldman Sachs analyst Thomas Berquist called Red Hat's valuation "rich" and raised the specter of pricing pressure on Advanced Server [since renamed Enterprise Linux AS]. Soundview Technology analyst Victor Raisys wrote in a Dec. 18 report: "We continue to believe that the current stock price reflects an overly optimistic view of Red Hat's ability to capitalize on the Linux opportunity." Based on estimates that it will earn 6 cents a share in its 2004 fiscal year, its forward price-earnings ratio is a lofty 92. Both Berquist and Raisys rate Red Hat underperform. (Stone 2003)

It will be interesting to see how competition will play out for Red Hat. This competition will come not only from Microsoft, but also from other Linux distributors, and potentially even Red Hat itself.

Outside Competition

Red Hat is not alone in its market however one would define it. They compete against other commercial Linux providers, not-for-profit institutions, and commercial software developers such as Microsoft.

Red Hat is not the only provider of commercial Linux solutions. In many ways, it competes with its partners such as IBM. IBM has shifted its focus to become a technology consulting business. As such, firms looking to build or switch an IT infrastructure often choose IBM as it can provide a "total package." IBM has a strategic relationship with Red Hat, but has invested over $1 billion in Open Source projects like Linux and Apache, the Open Source web server, and has plans to invest hundreds of millions more in the coming future. (IBM 2002) IBM may eat into Red Hat's consulting businesses. There are also other firms that focus on the development of Linux such as SuSE and Mandrake. These two companies are based in Europe and have the majority of their business there. SuSE has been more business focused while Mandrake has focused on individual users and ease of use. Both companies have a presence in the United States, and SuSE especially seems primed to enter into the business alongside Red Hat as a Linux business solution provider.

There is also the issue of another company using the same products but selling a different, more desirable service. Recently Sun Microsystems experimented with its own

distribution of Linux which was based on Red Hat Linux. Ultimately, they found it easier just to use Red Hat as their Linux distribution. Another firm, Terra Soft Systems, has adapted Red Hat to run on Apple Macintosh computers called Yellow Dog Linux. But for a few added features, it is Red Hat Linux, but resold by a different company. Red Hat has no legal claim over their own software, and leaves itself open to "poaching" by competitors yet to come.

An interesting alternative is that of a not-for-profit organization such as Debian. The Debian Project is a competing distribution of Linux that, as a non-profit, has different priorities than does Red Hat. In fact, Debian is considered to the "most free" version of Linux available. The Open Source Guidelines derive directly from the Debian Free Software Definition. Debian focuses on stability and has long product cycles. It also does not charge users for upgrades along its network. Debian has a system called "apt-get," which is remarkably simple and easy to use to upgrade computers running their distribution of Linux. Debian also has corporate support. HP sponsors the Debian Project and sells low end "blade" servers that run the Debian distribution of Linux. Because Debian does not look to earn revenue from upgrades the way Red Hat does, it has even less of an incentive to try and force costly upgrades on consumers.

Red Hat began selling its latest distribution of Linux, version 9, only a few months after version 8 was sold. This caused some uproar as it seemed as if it was shoehorning unnecessary upgrades on its clients and RHCEs. The RHCE certification is supposed to be good for two major versions, and a jump from 8 to 9 would be one such jump. There was a panic on Slashdot after the announcement and Red Hat said that it would change the scheme by which it would certify their RHCEs. This numbering

scheme is reminiscent of the tactics often associated with commercial software firms such as Microsoft. It has been seen as an attempt to boost revenue through changing its numbering scheme on consumers.

Microsoft is one of the most dynamic corporations. A company that built itself up on the sale value of software is not likely going to give up its business model and its livelihood without a fight. Microsoft might not have been the first company to take advantage of the Internet, but Internet Explorer is now the most popular web browser. Netcraft's latest survey shows that Microsoft Internet Information Server has roughly a quarter of the web server market even though it only runs on it own Microsoft Windows (2003).

Both Microsoft Windows and Linux have been gaining server market share at the expense of proprietary Unix systems. Microsoft has also been focusing on the market for larger servers and on stability and security, attributes with which Unix systems and Linux have been traditionally associated. Microsoft also has $40 billion in assets which are available to use against any and all problems.

Microsoft commissioned a survey recently from IDC which concluded that for many uses, Windows can offer a lower cost of ownership (TCO) than Linux offers. The argument is that while Linux is free, it often requires a greater amount of expertise to run Linux servers. Windows also has more software available to make running these systems easier for many tasks.

> The study concluded that because Microsoft had created more software tools for managing and updating Windows, the operating system would be 11% to 22% cheaper to run than Linux over a five-year period in four out of five different common computing tasks, such as sending files to printers and running security applications. Windows was more expensive when it came to serving up Web pages. (Greene 2003)

Microsoft has also tried other techniques to combat the threat of Open Source and Linux. I examined this more extensively in my study of the Microsoft Shared Source Initiative. Still, this TCO argument is possibly the strongest one to date. Jeff Cohen, the CIO at JetBlue, the startup airline credits using Microsoft Windows across the board as a major cost cutter. "By standardizing on one operating system and using other Microsoft software, the JetBlue CIO says he cut the company's technical staff by 50 percent" (Wilcox 2003). Through the use of Microsoft software, JetBlue was able to see major cost reduction improvements across the board.

Microsoft also has been using its extensive resources in order to move into higher end servers, embedded systems, and even into the entertainment market with the Xbox. They have a dominant hold on the desktop market, and are looking to introduce a number of innovative features in its next version of Windows. It would be truly foolhardy to discount Microsoft's competitive ability.

Final Questions

Ultimately, the question that I am looking to answer in this section is whether or not Red Hat is a viable business model. In the world of Open Source Software, revenues simply cannot be earned through a sale value model. Red Hat's model of improving and distributing Linux as a loss leader to support its other businesses seems to be working. While it has yet to show a significant profit, Red Hat has been showing solid growth. Ultimately Red Hat's future lies in the future of Linux. In its prospectus for its secondary offering, Red Hat outlines a number of very real risks to its bottom line. Here is a selection:

- "Our business may not succeed because open source software business models are unproven … No other company has built a successful open source business" (Red Hat 2000, 7).
- "Our reliance on the support of Linus Torvalds and other prominent Linux developers could impair our ability to release major product upgrades and maintain market share" (ibid).
- "We may lack the financial and operational resources needed to increase our market share and compete effectively with Microsoft, other established operating systems developers, software development tools developers and other service and support providers. In the market for operating systems, we face significant competition from larger companies with greater financial resources and name recognition than we have" (ibid, 12).
- "We could be prevented from selling our developing our products if the GNU General Public License and similar licenses under which our products are developed and licensed are not enforceable" (ibid, 17).

There is one risk which Red Hat leaves out: the risk of Linux becoming too good an operating system. The major advantage that Microsoft claims is that its ease of use and integration give it a lower TCO. If Linux becomes easy to use and integrate, firms and individuals may no longer require the level of service and support that Red Hat currently offers. While this would increase the size of the market for these services, Red Hat may very well have the incentive to keep Linux from becoming "too good." Microsoft now finds itself in a position where it has to convince its clients to upgrade their systems to newer versions of their software as their security, stability, and ease of use has improved. Red Hat might find itself in a similar position at a stage where they do not have $40 billion to spend on research and marketing. Even if these clients were to upgrade, Red Hat does not own any rights to Linux. Red Hat's clients could simply download the software freely from the Internet bypassing Red Hat and its Red Hat Enterprise Network altogether.

There has never been a business model quite like that of Red Hat. All the analogous cases seem to fall short of fully describing Red Hat's unique model. Netscape used its browser as a way of selling its server software, but ultimately failed as Microsoft entered the market. IBM has a business model which develops Open Source Software as a loss leader, but it sells hardware and overall business consulting which Red Hat does not. The idea of giving away razors in order to sell blades is another analogous model. However, Linux users do not need to continually buy new "blades" from Red Hat in order to use the software. Furthermore, new "blades" (or in this case upgrades) could be also downloaded at no cost. Network television gives away its programming and earns profits through advertising revenue, but viewers are subjected to the commercials whether they want to or not. Red Hat could not force either advertisements or its services onto its users. Additionally, in the coming age of video on demand and peer-to-peer networks, the network television model may not even be long term viable. Red Hat is in unchartered business waters.

This raises the issue of what is a viable business plan for Open Source Software development if the Red Hat model ultimately fails. Perhaps the development of the software will fall to individual and firm based user-developers and non-profits, such as Debian or the Apache Group, in conjunction with one another. Then firms may just sell software applications that run atop systems like Linux and Apache. Other options could be business plans such as IBM where IT consulting with hardware and software sales are a part of general business consulting. I believe Open Source Software could still flourish without Red Hat. Ultimately, the future of Red Hat remains uncertain, but its success or failure will foretell the future of intellectual property based businesses.

X. Conclusions

Working 1.0 Theory

After looking at the case studies and historical examples, some patterns emerge. Looking most closely at the case of Mozilla, there appear to be necessary prerequisites for innovation to occur under the Open Source paradigm. Some such prerequisites include what I call a "Working 1.0" version of the software.

By a "Working 1.0" version, I mean that in order for sufficient development to occur, there needs to be a working base of code. From this base of working code further innovation can spring. Apache was based on the already working and popular NCSA HTTP server. By using an adequate starting point, development can flourish. Mozilla presents a striking counterexample. Mozilla was, in the beginning, a mess of code that did not work. Only later, when the code was largely rewritten, that development started to flow in from the outside community.

I do not mean that it must literally be numbered version 1.0. The numbering system can be rather arbitrary. For instance, Apache did not have its 1.0 release until a year after the project started, but it was based on an already functioning base. Without a minimum of functionality, users will not take to the project and will not develop for it.

Microsoft Shared Source should prove an interesting case in this respect. Microsoft has popular and well functioning software that has a presence in all aspects of computing. Whether innovation from the community will be spurred on by Microsoft's opening of the code will yet be seen.

Viability of Open Source as a Software Development Paradigm

This thesis has been an examination of Open Source methods and whether they compose a viable paradigm for software development. Through review of academic works, historical comparison, news articles, and case examination, I conclude that it is indeed a powerful tool for development of high quality software products.

Linux and Apache have proven that, even in the absence of intellectual property protections, Open Source Software can achieve commercial quality. Network effects and positive feedback loops help OSS projects gain momentum and allow for these projects to be self sustaining. The Internet has facilitated Open Source's surpassing of previous instances of know-how trading, and the cross-industry importance of information technology has allowed Open Source methodology to pervade all areas of computing.

Still, Open Source is not a panacea for all software development problems. It also cannot always aggregate demand sufficiently to develop certain types of software. The examples of Mozilla and GUIs best illustrate this. When Netscape first released the source code to its Communicator product, there was very little outside support. Only after rewriting large portions of the code, providing documentation for the code, and providing innovation toolkits did the Mozilla project take off. It took a full four years for Mozilla to reach its 1.0 release. Open Source GUIs, such as KDE or GNOME, pale in comparison to the more mature and sophisticated user interfaces, such as Mac OS or Microsoft Windows. Because the market for these products is largely focused on users who lack the skills to be user-developers, strong intellectual property protection may be necessary to facilitate these kinds of software products.

Viability of Open Source as a Business Tool

Open Source can certainly be seen as an effective business tool. The advantages that Open Source presents to firms are not exclusive to software developing firms. Open standards allow purchasing firms to avoid lock in and can allow firms to standardize their systems across the board. This can make for drastic cost reductions. Hardware firms such as Dell use Linux in order to sell low cost servers. IBM has used Linux and Apache to undercut competition and restructure the server industry in its interest.

Open Source has even facilitated the creation of new business plans. Red Hat has a business plan that is focused on providing services around a series of software packages. IBM shifted its entire business focus to take advantage of the lower costs of that Open Source development offers. Being able to ignore a problem of an operating system, new technologies have emerged where Linux is embedded. A new era of cell phones, PDAs, set top boxes, and other emerging technologies will be based around the Open Source paradigm. As information technology revolutionizes more and more businesses, Open Source will further revolutionize information technology.

Final Thoughts

Open Source Software Development is a fascinating development paradigm for complex products which appeal to a discriminating, heterogeneous user base. The power of the Internet has allowed for know-how trading to be done on a massive scale with input from all corners of the earth. Ultimately, Open Source development is consistent with prevailing economic theory and economic history. The Open Source paradigm also shows us that many of the preconceived notions of strong legal protections being

necessary incentives for development may be incorrect. Put simply, Open Source is a viable and healthy method of developing high quality software and has a long and fascinating history ahead of it.

Glossary

Apache – Open Source/Free Software used to serve web pages over the Internet. Apache can run on either Unix systems or on Windows systems. According to the Netcraft survey, Apache is the number one web server in the world. Microsoft's IIS is the second most popular.

BSD – Berkeley Software Distribution. A free implementation of the Unix operating system developed by the University of California, Berkeley. It is released under what has been called the BSD license. The BSD license has virtually no restrictions on the use of its code and as a result most modern operating systems have some BSD code somewhere in their code.

GPL – General Public License. Developed by the Free Software Foundation, the GPL is a special type of Open Source/Free Software license. The GPL forces code that derives from other code covered under the GPL to also be distributed under the same license. This means that any code developed under the GPL will be free, and any other code that may borrow from the original will be also be free with the same stipulations. This license is often criticized as "viral" because that programs "infected" with GPL licensed code become GPL licensed code as well.

Halloween Documents – internal Microsoft memoranda. The first one was believed to have been first written on October 31, 1998. Because Oct. 31 is Halloween, they were dubbed the Halloween Documents by the media.

IT – Information Technology. Information technology is the term used to describe the computing infrastructure used in any given institution. This general term applies to the workstation computers, servers, printers, and anything else that generally falls under the umbrella of day to day technology used.

LGPL – Lesser General Public License. Like the GPL, the LGPL is an Open Source/Free Software license. The LGPL is less restrictive than the GPL. LGPL code can be integrated into closed software products, but any alteration made to the LGPL code must be released freely.

License – a term of use agreement between the software user and developer. These licenses have various levels of restrictiveness. The Microsoft's End User License Agreement places a number of restrictions on the kinds of use and on the number of computers on which an individual copy could be used. The BSD license, conversely, places virtually no restrictions on the use of the software.

Linux – Linux is easiest to describe as a Unix variant operating system. Originally developed by Linus Torvalds while a graduate student at the University of Finland, he now works for Transmeta, a semiconductor firm in California. Linux is a registered

trademark of Linus Torvalds. Linux is licensed under the GPL, and has a huge following of users and developers all over the world.

Mainframe – A mainframe can be thought of as a large computer that lies somewhere in the back room. They are typically very powerful and run large software applications such as large databases. The word mainframe is an anachronism from the pre-personal computer days when computing power had to be held in a single location due to expense.

Middleware – nebulous term that refers to certain types of software. It means that some software lies below it, such as an operating system, and others lie on top of it, like a database. Apache fits into this category as it lies atop the operating system and below the web pages being served by it.

Server – A large, often powerful computer that "serves" web pages and other applications such as databases to users like web page viewers. This is a more contemporary term, but a server is not necessarily a super powerful computer the way a mainframe was. An older computer can often be used to serve web pages although large corporations with many clients who wish to view their web pages need powerful computers to deal with the heavy loads demanded.

Unix – An operating system for servers. Variants of Unix run just about all of today's servers. Linux, IBM's AIX, HP's HP-UX, and Sun's Solaris are examples of Unix variants. Linux was the first major variant of Unix to run on a typical personal computer, the effectively open x86 architecture.

x86 – A microprocessor architecture standard. This refers to the architecture that is found on the majority of personal computers. The term "x86" comes the line of microprocessors from Intel including the 386 and 486. Its later chips, the Pentium series, are considered x86 processors, and other competing microprocessors from firms such as AMD, fit under the x86 umbrella.

Bibliography

Apache Software Foundation. (2003), http://apache.org/ (19 March 2003)

Apple Computer, Inc. (2001), "Darwin: Mac OS X's Core OS." 8 March 2001 http://developer.apple.com/pdf/mactech_darwin.pdf (5 May 2003)

Apple Computer, Inc. (2002), "Darwin – Open Source." http://developer.apple.com/darwin/ (3 May 2002).

Bessen, James. (2001), "Open Source Software: Free Provision of Complex Public Goods." *Working Version May 2001*. http://www.researchoninnovation.org/opensrc.pdf (5 May 2003).

Bessen, James and Eric Maskin. (1999), "Sequential Innovation, Patents, and Imitation." *Working Paper 11/99*. http://papers.ssrn.com/paper.taf?abstract_id=206189

Bezroukov, Nikolai. (1999), "A Second Look at the Cathedral and the Bazaar." *First Monday*. Vol. 4. No. 12. 9 Dec 1999. http://firstmonday.org/issues/issue4_12/bezroukov/index.html (5 May 2003)

Dvorak, John C. (2002), "Upstarts Attack Microsoft Slackers." *PCMag.com* 26 Nov 2002 http://www.pcmag.com/article2/0,4149,715464,00.asp (5 May 2003)

Field, Jeff. (2002), "Review: TiVo Series2 Personal Video Recorder." *NewsForge*. http://newsforge.com/article.pl?sid=02/04/23/0259205&mode=thread&tid=7 (24 April 2002).

Free Software Foundation. (1991), "GNU General Public License." Version 2, June 1991 http://www.gnu.org/copyleft/gpl.html (5 May 2003)

Free Software Foundation. (1999), "GNU Lesser General Public License." Version 2.1, Feb 1999. http://www.gnu.org/copyleft/lesser.html (5 May 2003)

Free Software Foundation. (2002), "GNU's Not Unix!" http://www.gnu.org (4 April 2002).

Franke, Nikolaus and Eric von Hippel. (2002), "Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software." *MIT Sloan School of Management Working Paper*. Jan 2002. http://opensource.mit.edu/papers/rp-vonhippelfranke.pdf (5 May 2003)

Fried, Ian. (2002), "Ads claim adding HP is a Plus." *CNET News.com*. 17 Nov. 2002. http://news.com.com/2102-1001-966093.html (17 Nov. 2002).

Greenberg, Jonah. (2000), "Linux in China: Not Ready for Prime Time." *Salon.com*. 9 Aug 2000. http://www.salon.com/tech/feature/2000/08/09/linux_china/ (5 May 2003)

Greene, Jay. (2003), "Pecked by Penguins." *Business Week*. 3 March 2003 http://www.businessweek.com/print/magazine/content/03_09/b3822610_tc102.htm?tc&sub=03linux (25 April 2003)

Harhoff, Dietmar, Joachim Henkel and Eric von Hippel. (2000), "Profiting from Voluntary Information Spillovers: How Users Benefit by Freely Revealing Their Innovations." *MIT Sloan School of Management WP#4125*. 25 July 2000. http://opensource.mit.edu/papers/evhippel-voluntaryinfospillover.pdf (5 May 2003)

HP. (2002), "ProLiant BL p-Class: Product Overview." http://www.compaq.com/products/servers/proliant-bl/p-class/description.html (17 Nov. 2002)

IBM. (2002), "IBM Linux Portal - Linux at IBM." http://www-1.ibm.com/linux/index.shtml (17 Nov. 2002).

Johnson, Justin. (2001), "Economics of Open Source Software." 17 May 2001 http://opensource.mit.edu/papers/johnsonopensource.pdf (5 May 2003)

Kary, Tiffany and Stephen Shankland. (2002), "Sun details plans for Linux servers." *CNET News.com* 7 February 2002. http://news.com.com/2102-1001-831618.html (7 February 2002)

Lancashire, David. (2001), "Code, Culture and Cash: The Fading Altruism of Open Source Development." *First Monday*. Vol 6 Issue 12. 19 Nov 2001. url: http://www.firstmonday.org/issues/issue6_12/lancashire/index.html

Leonard, Andrew. (1999), "Open Season." *Wired Magazine*. 7.05 – May 1999 http://www.wired.com/wired/archive/7.05/open_source_pr.html (6 May 2003)

Leonard, Andrew. (2003), *Salon Free Software Project*. http://www.salon.com/tech/fsp/index.html (3 May 2003)

Lemos, Robert. (2002), "Sun plays safe with Linux device." *CNET News.com*. http://news.com.com/2102-1001-966257.html (18 November 2002)

Lerner, Josh and Jean Tirole. (2000), "The Simple Economics of Open Source." 29 Dec 2000. http://opensource.mit.edu/papers/Josh Lerner and Jean Triole - The Simple Economics of Open Source.pdf (5 May 2003)

Lerner, Josh and Jean Tirole. (2002), "The Scope of Open Source Licensing." 19 Nov 2002. http://opensource.mit.edu/papers/lernertirole2.pdf (5 May 2003)

Lettice, John. (2002), "Windows costs less than Linux. A bit. Sometimes - MS study." *The Register*. 3 Dec 2002 http://www.theregister.co.uk/content/4/28408.html (5 May 2003)

Linuxiso.org. (2001), "An Introduction to Linux." http://www.linuxiso.org/introtolinux.html (5 May 2003)

McGowan, David. (2000), "Legal Implications of Open-Source Software." http://papers.ssrn.com/paper.taf?abstract_id=243237 (3 May 2003)

Microsoft Corporation. (2003a), "Microsoft Shared Source." Mar. 2003. http://www.microsoft.com/ resources/sharedsource/default.mspx (3 May 2003)

Microsoft Corporation. (2003b). "Windows Server 2003 Home" http://www.microsoft.com/windowsserver2003/default.mspx (11 Mar. 2003)

Microsoft Corporation. (2003c), "Microsoft Windows Services for UNIX Product Overview" 3 May 2003. http://www.microsoft.com/windows/sfu/productinfo/overview/default.asp (3 May 2003)

Microsoft Corporation. (2003d), "Microsoft Windows XP Professional." http://shop.microsoft.com/Referral/ProductInfo.asp?siteID=10798&typeID=1 (27 Apr. 2003)

Mitchell, Russ. (2001), "Open War." *Wired Magazine*. 9.10 – Oct 2001 http://www.wired.com/wired/archive/9.10/linux_pr.html (6 May 2003)

Mockus, Audris, Roy T. Fielding and James D. Herbsleb. (2000), "Two Case Studies of Open Source Software Development: Apache and Mozilla." June 2000.

Moody, Glyn. (1997), "The Greatest OS That (N)ever Was." *Wired Magazine*. 5.08 – Aug 1997 http://www.wired.com/wired/archive/5.08/linux_pr.html

Mozilla Organization, The. (2003), http://mozilla.org (3 May 2003)

Mundie, Craig. (2001), "Microsoft's Views on Open Source and GPL Licensing" *republished e-mail*. 26 July 2001. http://use.perl.org/article.pl?sid=01/07/26/2341206&mode=thread&threshold= (3 May 2003)

NASA. (2003), "Beowulf Project at Goddard Space Flight Center" http://beowulf.gsfc.nasa.gov/ (1 May 2003)

Netcraft. (2003), "Netcraft Web Server Survey." 13 April 2003. http://news.netcraft.com/archives/web_server_survey.html (3 May 2003)

Netscape Communications Corp. (1998), "NETSCAPE ANNOUNCES MOZILLA.ORG." 23 February 1998. http://wp.netscape.com/newsref/pr/newsrelease577.html (3 May 2003)

Nuvolari, Alessandro. (2003), "Open Source Software Development: Some Historical Perspectives." Jan 2003. http://opensource.mit.edu/papers/nuvolari.pdf (5 Jan. 2003)

Open Source Development Network (OSDN), "SourceForge.net: An Overview of the SourceForge.net User Community" http://sourceforge.net/docman/display_doc.php?docid=9331&group_id=1 (3 May 2003)

Open Source Initiative, The. (2003), "Halloween Documents." http://www.opensource.org/halloween/index.html (3 May 2003)

Orlowski, Andrew. (2001), "Sun assumes full control of iPlanet." *The Register* 24 Aug 2001. http://www.theregister.co.uk/content/4/21251.html

O'Reilly, Tim. (1999), "How the Web was Almost Won." *Salon.com* 16 Nov 1999. http://www.salon.com/tech/feature/1999/11/16/microsoft_servers/print.html

O'Reilly, Tim. (2000), "Ask Tim." O'Reilly and Associates. 20 Jan 2000. http://www.oreilly.com/ask_tim/asp_php.html

O'Reilly Network. (2001), "Shared Source vs. Open Source: Panel Discussion" 9 Aug 2001. http://linux.oreillynet.com/lpt/a/1135

Patrizio, Andy. (2001), "Microsoft's Shared Source Record." *O'Reilly Network*. 25 Mar 2001 http://www.ondotnet.com/lpt/a/1662

Pavlicek, Russell C. (2000), *Embracing Insanity: Open Source Software Development*. Indianapolis: Sams Publishing.

Raymond, Eric S. (2001), *The Cathedral and the Bazaar*. Cambridge: O'Reilly.

Red Hat, Inc. (2000), "Prospectus ." 3 Feb 2000.

Red Hat, Inc. (2002), "Red Hat Annual Report 2002."

Red Hat, Inc. (2003a), "Global Learning Services." http://www.redhat.com/training/ (28 Apr. 2003)

Red Hat, Inc. (2003b), "Red Hat Enterprise Linux." https://www.redhat.com/software/rhel/ (3 May 2003)

Red Hat, Inc. (2003c) "Red Hat Enterprise Network." https://www.redhat.com/software/rhen/ (27 Apr. 2003)

Red Hat, Inc. (2003d), "What is Red Hat's Business Model?" http://www.redhat.com/about/mission/business_model.html (27 Apr. 2003)

Reis, Christian Robottom and Renata Pontin de Mattos Fortes. (2002), "An Overview of the Software Engineering Process and Tools in the Mozilla Project." http://opensource.mit.edu/papers/reismozilla.pdf (8 Feb 2002)

Rohde, Laura. (2002), "EU clears IBM-PwC Consulting deal." ITWorld.com http://www.itworld.com/Man/2713/020923ibmpwc/pfindex.html (23 September 2002)

Shankland, Stephen. (2002a), "IBM, Dell nab more server turf." *CNET News.com* 13 Nov. 2002. http://news.com.com/2102-1001-965697.html (13 Nov. 2002)

Shankland, Stephen. (2002b), "IBM extends its server lead." *CNET News.com* 22 Nov. 2002 http://news.com.com/2102-1001-966926.html (22 Nov. 2002)

Shankland, Stephen. (2002c), "Sun's Linux PC cheaper, McNealy says." *CNET News.com* 18 Sep. 2002. http://news.com.com/2102-1001-958487.html (18 Sep. 2002)

Slashdot.org. (2001), "New Microsoft Feature: Planned Obsolescence." 8 May 2001. http://slashdot.org/article.pl?sid=01/05/08/1153245&mode=thread&tid=109 (3 May 2003)

Slashdot.org. (2003a), "Red Hat 9 To Be Released March 31." 24 March 2003 http://slashdot.org/article.pl?sid=03/03/24/1947249&mode=thread&tid=110&tid=163 (3 May 2003)

Slashdot.org. (2003b), "Shared Source vs. Open Source." 10 Feb. 2003. http://slashdot.org/article.pl?sid=03/02/10/1213257 (3 May 2003)

Spooner, John G. (2001), "Linux slips off Dell computers." 2 Aug 2001. http://news.com.com/2102-1001-271006.html (3 May 2003)

Stone, Amey. "Red Flags for Red Hat." *Business Week*. 3 March 2003. http://www.businessweek.com/print/magazine/content/03_09/b3822615_tc102.htm?tc&sub=03linux (3 May 2003)

Sun Microsystems, Inc. (2002), "Sun LX50 Server Overview" http://www.sun.com/servers/entry/lx50/ (23 Nov. 2002)

Sun Microsystems, Inc. (2003a), "Sun History." http://www.sun.com/aboutsun/coinfo/history.html (8 Mar. 2003)

Sun Microsystems, Inc. (2003b), "Sun ONE Web Server Enterprise Edition." http://store.sun.com/catalog/doc/BrowsePage.jhtml?cid=64488 (11 Mar. 2003)

TiVo, Inc. (2003), "Customer Support." http://www.tivo.com/linux/index.html (3 May 2003)

Wendt, Steve. (2003) "OS/2 Warp History" http://www.os2bbs.com/os2news/OS2Warp.html (3 May 2003)

von Hippel, Eric. (2002), "Horizontal Innovation Networks – By and For Users." June 2002 http://opensource.mit.edu/papers/vonhippel3.pdf (5 May 2003)

von Hippel, Eric and Georg von Krogh. (2002), "Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science" 30 April 2002

Wilcox, Joe. (2003), "Helping JetBlue see black." *CNet News.com*. 23 April 2003 http://news.com.com/2102-1082-997868.html (23 April 2003)

Yahoo! Finance. (2003) "Red Hat, Inc." http://finance.yahoo.com/q?s=rhat&d=t (3 May 2003)

Yoffe, David B. and Mary Kwak. (2001), "The Browser Wars, 1994-1998." *Harvard Business School Case Study*.

Zawinski, Jamie. (1999), "Resignation and Postmortem." 31 Mar. 2001 http://www.jwz.org/gruntle/nomo.html (3 May 2003)

Zawinski, Jamie. (1994), "The Netscape Dorm." http://www.jwz.org/gruntle/nscpdorm.html (3 May 2003)

Additional assistance was received from Prof. Anne P. Carter of Brandeis University, Prof. Eric von Hippel of MIT Sloan School of Management, and Benjamin Cox of the Linux Fund.