An Economic Survey of Open Source Software

Kevin Penner

Contents

1 Introduction **3**

2 What is Open Source Software? **4**

3 What Motivates Programmers **6**

4 A New Development Structure **8**

5 Economic Issues of OSS **10**

5.1 Costs and Benefits of OSS 11

5.2 Code as a Public Good and the Principal-Agent Problem 13

5.3 Asymmetric Information and Transparency in Software Development 15

5.4 Free Rider Dilemma and OSS 15

6 Conclusion **16**

List of Figures

1 Linux's Increasing Market Share 18

2 Firefox's Increasing Market Share 18

3 Apache's Market Dominance 18

1 Introduction

When looking for software, most computer users simply browse their local electronics store. These consumers usually need no more than a copy of Microsoft Windows and Office for all of their computing needs, but these products are not cheap. The basic version of Windows Vista costs around $100, and the home version of Microsoft Office costs $150. In some cases, software can cost more than a computer. For example, Adobe Photoshop costs around $700!

Given the high cost of some software, it may surprise consumers that high quality, free alternatives exist. One could replace Windows Vista with Ubuntu Linux, Microsoft Office with OpenOffice and Adobe Photoshop with Gimp. These free programs are part of an open source software (hereafter OSS) movement that has captured roughly six percent of the world market for software [16]. Although this market share seems relatively small, OSS has led to a savings of roughly $60 billion per year for consumers [16].

However, most computer users are skeptical of OSS. Is the code stable? Is the software easy to use? Does the software have a virus? Is using this software legal? The phrase Linux elicits visions of computer science geeks and fourteen year old hackers in Russia. Despite being a complete operating system developed by thousands of computer enthusiasts for free, Linux is not seen with awe but fear.

Popular skepticism aside, OSS offers great benefits. Not only is it free, OSS is often better than software sold in stores [5]. Consider the following: in studies performed at Stanford and Carnegie Mellon, researchers found that the Linux 2.6 kernel had only 0.17 bugs per 1000 lines of code while proprietary software generally had 20-30 bugs per 1000 lines [4].

Having matured to this point, open source software is becoming more important in the information technology industry. At first glance, OSS development seems to be an economic anomaly. Why do rational programmers freely share valuable software code? Are they altruistic or merely as crazy as they often appear?

This paper is an overview of some of the economic issues surrounding open source software. The paper is organized as follows. In the next section, I establish some basic principles of OSS. I look at what motivates computer programmers to participate in the third section. Given the less traditional motivations of OSS programmers, I discuss an emerging organizational structure in section four. Viewing OSS as a tool, not an industry, I then identify some of the costs, benefits and economic properties of OSS in the fifth section. Section six concludes the paper.

2 What is Open Source Software?

Making the distinction between source code and compiled code is extremely important. A computer program is a set of code that gives commands to a piece of computer hardware. Most code is written in a programming language that humans can understand and edit. This code is known as *source code*. However, for source code to perform a function, it must be converted into machine language that hardware can process. This *compiled code* is a set of 1's and 0's that is almost impossible for humans to interpret.

Keeping this distinction in mind, I divide software into two broad categories: proprietary software and OSS. For the purposes of this paper, I use the term proprietary software to

refer to software that is closed source. In this case, users only get the compiled code. The source code is not available to view, so these programs are nearly impossible for users to edit. Usually, proprietary software is the legal property of one party like Microsoft.

OSS is significantly different. The most important characteristics of OSS are [17]:

- The software license shall not require a royalty or other fee for such sale. The programs are free for all to use.
- The software must include the source code. This makes OSS radically different than proprietary software like Microsoft Office. Anyone who purchases Microsoft Office can only use the program, not inspect it.
- The license must allow the code to be modified and must allow the new works to be distributed under the same terms as original software. The programs remain free to use and enhance.

Basically, OSS is software where the source code is freely and publicly available. Anyone that uses open source software can modify the code. This allows users to expand functionality of the program and tailor it for their own use.

Open source is a unique development methodology. Since code can be modified and redistributed easily over the internet, thousands of programmers, not a single company, develop a single piece of software (see section 4). For example, "only about two percent of the current Linux kernel has actually been written by [Linus] Torvalds," the catalyst behind the Linux project [13]. Given the dependence of OSS on individual contributions, it is important to understand why individuals contribute to open source projects.

3 What Motivates Programmers

Lerner and Tirole [12] argue that software programmers perform a classic cost-benefit analysis before choosing to invest their time in a project. Studying the motivations of these programmers is important because it is not obvious why technically skilled people would contribute to unpaid projects.

Instead of working on open source projects, programmers could be getting paid for their services or focusing on other work. This wasted time presents a significant opportunity cost for programmers (and possibly their employers). The 'lost' compensation for "the existing base of [OSS] software [has] a lower bound of about 131,000 real person-years of effort... [This work is worth] at least €800 million each year" [22].

Nonetheless, there are quite a few compensating incentives for programmers to work on open source projects. Arguably, the main benefit a programmer derives from an open source project is an increase in the programmer's reputation. An important characteristic of OSS development is giving credit to an individual for code he contributed. Anyone can find out exactly what code a programmer contributed to a project, what bug he fixed.[1] By setting individuals apart like this, OSS development allows a programmer to send signals about his ability.

The concept of signaling is key to understanding the economic incentives of programmers. Before discussing benefits of the signals, it is important to note that signaling is more effective

[1]This information can be found by checking things like comments in the source code and the changelog file that accompanies the code.

in the open source environment than the proprietary environment because if the performance is more visible, the signaling incentive is stronger [12]. Writing code for OSS is a better performance measurement since outsiders can observe the individual's precise contributions. However, outsiders can only inexactly observe an individual's contributions to proprietary code since individual contributions are hidden.

OSS allows a programmer to signal his ability to a large, relevant audience. This peer recognition increases a programmer's reputation. The personal benefits from this recognition are two-fold. First, like many others, programmers get ego gratification through peer recognition [12]. Second, these reputation benefits can be turned into monetary rewards in the form of future job offers.

To back up this signaling argument, consider the following analogy from Lee, Moisa and Weiss [11]. A Ph.D. student has incentives similar to those of an open source programmer. By openly publishing research, this student sends a signal that sets him apart from other college graduates. However, to send this signal he has to forgo immediate rewards like wages. OSS programmers face a similar tradeoff. To send a signal, they cannot restrict access to their code. Since commercial software is usually closed source, programmers probably cannot receive wages and allow free access to their product. Thus, an OSS programmer must give up wages to send a signal much like a Ph.D. student does.

Weber [23] notes that reputation might not be the strongest incentive to contribute. If it was, then there would probably be more ambition for leadership, which would lead to more deviations from development paths. Projects would fragment before completion.

However, there are additional incentives for individuals to participate in OSS development. Many programmers desire to participate in a community that they identify with [20]. Programmers also get the chance to enhance their coding ability, fix bugs and customize software. Finally, programmers get to work on projects that are seen as creative and fun [12].

Programmers only help on projects that interest them, and this interest is an extremely important motivational tool. Since open source developers are not subject to the demands of a manager, programmers are free to try and add new features to programs without having to worry about things like marketability. These added features might be available from OSS before proprietary software. For example, iPodLinux allowed iPod owners to play crude videos on their iPods before Apple came out with this functionality!

4 A New Development Structure

The collaborative nature of OSS development is a new organizational innovation. In the mid-1900s, Ronald Coase identified two main ways in which economic production is organized: individuals act as employees in firms and as individuals following price signals in markets. Surprisingly, OSS production seems to fall under neither of these organizational categories [3].

The boundaries that exist at the firm level do not exist in OSS development. OSS development has shown the viability of a new organization of production called "commons-based peer production" [3]. Here, individual programmers collaborate on large scale projects

with diverse motivations that, according to section three, have little to do with a manger's demands or immediate monetary incentives.

A distinctive feature of this type of production is the decentralized nature of information gathering and exchange. Eric Raymond compares the OSS development process to "a great babbling bazaar of differing agendas and approaches" [19]. Programmers across the globe coordinate almost exclusively online via e-mail and forums. The key to managing this form of development is modular design [23]. This keeps programs small and focused, but these small programs are easily combined to perform more complex tasks.

The decentralized nature of OSS development might seem chaotic, but it has distinct advantages over the traditional firm/market model since information (code) is what is being produced [3]. Whereas

> proprietary software development tends to proceed at a pace determined by the least productive contributor ... OSS software development fully exploits the intelligence of the community and thus proceeds at a pace determined by the most productive member of the community [20].

This surprising comment warrants further explanation. The claim is that the OSS style of production is efficient at identifying and matching willing human capital to specific projects that need work ("exploiting the intelligence of the community"). This arises from the following facts. First, in OSS development, a large number of programmers searching for new projects are able to interact with a large number of resources. These resources are things like wikis, forums, mailing lists and other programmers. Second, since OSS places no limi-

tations on access to code that needs to be developed, openness of information helps allocate human capital efficiently. Individuals have the information needed to select and access an interesting project. This lowers organizational costs since personnel and information are less limited than under proprietary development. Additionally, OSS development drastically lowers transaction costs[2] of a software project [3]. Not only is commons-based peer production a new style of production, it is an efficient style of production.

OSS has no enforcing agency, no set leadership. A programmer's authority is based on the responsibilities he has successfully taken on. This could lead to problems. There is nothing to stop programmers from taking a project and "forking" it.[3] As was stated earlier, being a project leader has distinct signaling benefits for programmers. However, forking is not the norm in OSS development. Weber [23] attributes this to "developers think[ing] of themselves as trading innovation for others' innovation, [and] they want to do their trading in the most liquid market possible. Forking would only reduce the size and thus the liquidity of the market."

5 Economic Issues of OSS

It is important to look at OSS from the right perspective. Since it is essentially free, OSS does not present any profit opportunities for companies outside of support services. OSS should *not* be viewed as an industry. Instead, OSS should be viewed as a tool. When

[2]Transaction costs are costs associated with defining and enforcing property and contract rights.

[3]A project is forked when developers take source code from one piece of software and start independent development on another similar application.

discussing the economic impact of open source software, keep in mind

> Microsoft is a tool-maker, and the effect of the tool-maker on the economy is tiny next to the economic effect of all of the people who are enabled by the maker's tools. The secondary economic effect caused by all of the people and businesses who use an enabling technology is greater than the primary economic effect of the dollars paid for that technology. And of course the same is true for Open Source software [18].

The main economic contribution of software is that it enables companies to do business more efficiently. For example, OSS like Apache, which has a 50% share of the web server market, and Sendmail enable a vast amount of e-commerce. Software should be viewed as a means, not an end.

5.1 Costs and Benefits of OSS

"There is no such thing as a free lunch" is a mantra of the economics profession. Although it is often the case that OSS is free, the use of such software has costs. The most significant is software support. Red Hat Chief Executive Jim Whitehurst notes that "a lot of [his] customers... have said that the support costs [for OSS] are really quite high" [21]. This arises because OSS is historically harder to use than proprietary software. OSS is not usually designed for inexperienced users. Whereas commercial forms can afford to spend money on graphical user interfaces and documentation, open source projects usually forgo user friendliness, focusing instead on creating stable, powerful programs. Companies that

decide to use Linux machines might save money on software but spend more on training and support.

However, free support for OSS does exist, mainly in the form of user-to-user assistance. Lakhani and von Hippel studied this type of free assistance for the Apache web server software. They found that most questions posted by users were answered quickly, and most of these answers were valuable to the question posters [10]. They predict that only one-fourth of posted questions do not receive an answer. For comparison, in 2008 Dell's computer support, not a free service, received a 75/100 rating in customer satisfaction [8].

Other important drawbacks to OSS are the network externalities users face. For example, the widespread use of the Microsoft Windows operating system encourages firms to produce complementary software. This is why there is a disproportionately large collection of Windows software like Microsoft Word on the market. This collection increases the value of the Windows operating system, furthering the use of Windows and its related software. A drawback of OSS is the unavailability of similar products being produced for Windows. If every other business is using Microsoft Word, a new business is likely to use Word as well.

There are offsetting benefits to using OSS. The most obvious is a drastic decrease in software costs. If a company needs an office productivity suite, OpenOffice costs the company nothing whereas Microsoft Office could easily cost thousands of dollars. Additionally, the incremental cost of debugging or adding a desired feature to OSS is generally much smaller than the cost of developing software from scratch.

Not only do companies have an incentive to use OSS, firms have an incentive to release

code that they are entitled to keep private since it will become part of a publicly supported code base [7]. This can make the code less expensive to maintain. Apple is a prime example of a company that believes this. They have made "open source development a key part of ... ongoing software strategy" [2].

Most importantly, OSS tends to have better written code than proprietary software (discussed in sections 5.2 and 5.3). This advantage implies that future costs of maintaining OSS are lower and that OSS is more stable (see introduction). Furthermore, the "release early, release often" belief in development keeps software functionality up to date. Although it takes time for software to become stable enough for businesses to use, "open source projects typically have a feedback and update cycle that is an order of magnitude faster than commercial projects" [23].

5.2 Code as a Public Good and the Principal-Agent Problem

OSS code is claimed to be more elegant than proprietary software code [9]. This claim can be justified by treating well written code as a public good.[4] Code is considered elegant if it is organized well and thoroughly explained. Although it involves a heftier time commitment, the obvious benefit of elegant code is the decreased future cost of modifying the program. Anyone who has ever had to debug software knows that obfuscated code is extremely difficult to deal with.

Elegant code is a public good in an informative sense. If more people can understand

[4]OSS is considered a public good since it is non-excludable (it can be distributed quickly and easily) and since one person's use of the software does not reduce its availability. See [9].

the code, then the code is more valuable. It is easier to modify, debug and extend. In the open source environment, there is a high incentive to produce clear, understandable code. Since OSS is developed collaboratively by hundreds or thousands of programmers all over the globe, elegant code is a necessity. This peer review system helps lead to the production of elegant code [9].

Proprietary software does not provide this public good because there is a principal-agent problem in proprietary software development [9]. Suppose the manager at a software firm wants a piece of software written to perform a certain function. The manager has to motivate his programmers to write the best code with respect to current and future profits accrued from the software. A programmer can spend a lot of effort commenting and organizing code so that future modifications will be easy, or the programmer can spend little time writing comments and produce unclear code. Unfortunately, closed source programmers face few repercussions for writing ugly code. A programmer knows he will likely not deal with the same code in the future, and it will be hard for an angry manager to trace poor code back to the individual [9]. Basically, updating the code will be another programmer's problem.

Since these programmers do not have to deal with their code in the future, and since it is costly to them to make the code easier for others to read, proprietary programmers have little incentive to expend a lot of effort writing elegant code. As long as the current code works reasonably well, the programmer is compensated and in the clear.

5.3 Asymmetric Information and Transparency in Software Development

Closed source software is a fantastic example of an asymmetric information problem. In the open source environment, code is readily available. The quality is easy for programmers to check and report, so the transparency of OSS lessens the information gap between the programmer and the user.

However, the developer of closed source software knows a lot more than the user about the stability and clarity of the code. Clarity of code can be interpreted in terms of Akerlof's classic lemon problem [1]. Upkeep of code can be viewed the same way as upkeep of a car: if code is better commented and organized, then the future cost of updating the code is lowered [9]. If the buyer and seller valuations for quality and shoddy software are significantly different, then it is possible that only 'lemon' code will be produced proprietarily [9].

This is not to say proprietary software does not work. Rather, it implies the source code is far from elegant although, by design, there are not many ways to evaluate this claim. The arguments in this and the previous section are simply an attempt to justify the claim (see section 5.1) regarding the possible superiority of OSS code over proprietary software code. Hopefully, some suspicions about the quality of OSS have been allayed.

5.4 Free Rider Dilemma and OSS

OSS presents ample opportunities to encounter the free-rider problem. Since programs are both free and readily available, it would be easy for individuals to take advantage of the

software without helping develop or promote it. However, since using OSS does not diminish a market or use a scarce resource, the traditional free-rider problem does not develop [18].

If a programmer helps develop a piece of software, he is obviously deriving some personal benefit. One source of this benefit is emotional fulfillment that comes from others using the software. It would be less satisfying to develop something no one uses, so users add to a program's value simply by using it. Thus, neither users nor contributors should be considered free-riders.

6 Conclusion

The development of OSS is an interesting economic case study. Even without the structure of a firm or monetary incentives, the open source paradigm is able to produce high-quality, powerful tools.

This paper has examined the basics of the open source development process. Individuals are not irrational for giving up their time to contribute to free software projects, for open source programmers perform a type of cost-benefit analysis before contributing to a project. Returns like increased peer recognition and enhanced code motivate programmers to participate. On the organizational level, although the open source process is decentralized, the ability of the process to exploit the intelligence of the programming community makes it an efficient way to design software.

This paper also examined the benefits of using OSS. Although OSS is generally not as user friendly as proprietary software, OSS is cheaper to maintain, more elegant and more

stable.

Despite these advantages, OSS is not going to replace the commercial software industry, only supplement it. Arguably, it targets too specific a subset of consumers. Nonetheless, the use of OSS is slowly increasing (see Figures 1-3). Hundreds of thousands of government computers across the globe run Linux and OpenOffice as replacements for Microsoft Windows and Microsoft Office [14] [15]. OSS also enables thousands of businesses and is an important source of technological advancement. The economic impact of OSS already measures in the billions of dollars. It would be beneficial to see development and usage continue to grow.

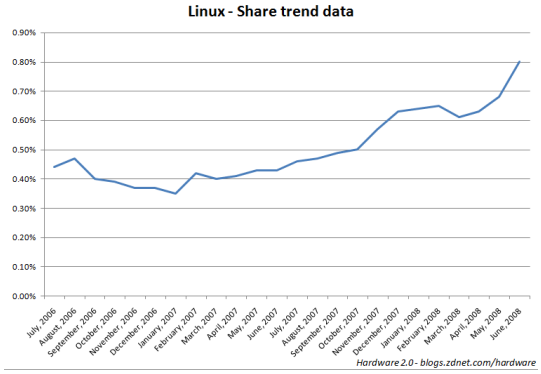

Figure 1: Linux's Increasing Market Share

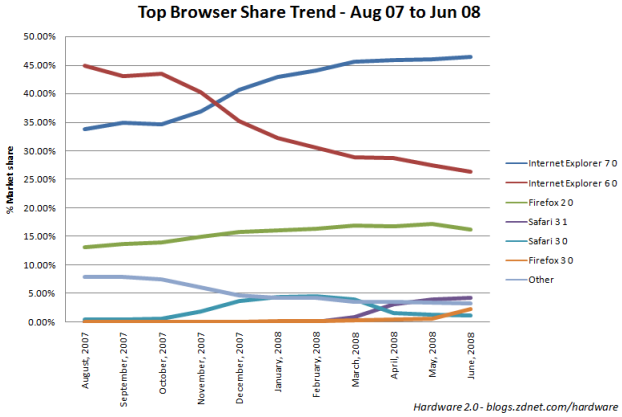

Figure 2: Firefox's Increasing Market Share

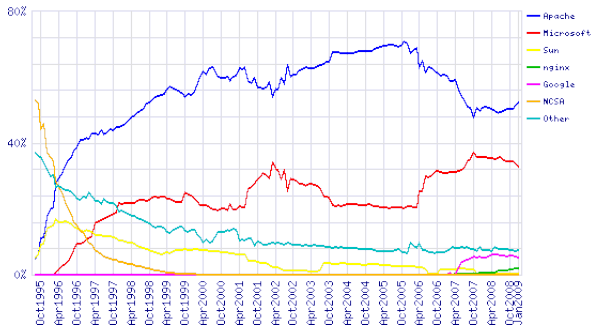

Figure 3: Apache's Market Dominance

References

[1] Akerlof, George. (1970, August). The Market for Lemons: Quality Uncertainty and the Market Mechanism. *The Quarterly Journal of Economics, Vol. 84, 488-500.*

[2] Apple Inc. (2009). *Open at the Source.* Retrieved from http://www.apple.com/opensource/.

[3] Benkler, Yochai. (2002). Coase's Penguin, or, Linux and the Nature of the Firm. *The Yale Law Journal, Vol. 112.*

[4] Delio, Michelle. (2004, December). *Linux: fewer bugs than rivals.* Retrieved from http://www.wired.com/software/coolapps/news/2004/12/66022.

[5] The Economist. (2000, April 13). *Open sesame.* Retrieved from http://www.economist.com/business/displaystory.cfm?story_id=E1_PPNQJD.

[6] Green, Eric Lee. (2002, December). *Economics of Open Source Software.* Retrieved from http://badtux.org/home/eric/editorial/economics.php.

[7] Hawkins, Richard. (2002, November 12). The Economics of Open Source Software for a Competitive Firm. *Kluwer Academic Publishers.* Retrieved from http://opensource.mit.edu/papers/hawkins.pdf.

[8] Keizer, Gregg. (2008, August 19). *Apple Clobbers Competition in Customer Satisfaction.* Retrieved from http://www.pcworld.com/businesscenter/article/150027/apple_clobbers_competition_

in_customer_satisfaction.html.

[9] Konovalov, Zoe. (2002, November). *The Economics of Open Source Software.* Retrieved from the Federal Trade Comission Web site: www.ftc.gov/os/comments/intelpropertycomments/konovalovzoe.pdf.

[10] Lakhani, Karim and von Hippel, Eric. (2002, June). How Open Source software works: "Free" user-to-user assistance. *Research Policy.* Retrieved from opensource.mit.edu/papers/lakhanivonhippelusersupport.pdf.

[11] Lee, Samuel, Moisa, Nina and Weiss, Marco. (2003, March 21). Open Source as a Signalling Device - An Economic Analysis. Retrieved from http://opensource.mit.edu/papers/leemoisaweiss.pdf.

[12] Lerner, Josh and Tirole, Jean. (2002, June). Some Simple Economics of Open Source. *The Journal of Industrial Economics, Vol. L, 212-220.*

[13] *Linus Torvalds: A Very Brief and Completely Unauthorized Biography.* (2006, January). Retrieved from the Linux Information Project Web site: http://www.bellevuelinux.org/linus.html.

[14] *Linux in Government.* (2008). Retrieved from http://www.linux.org/info/linux_govt.html.

[15] *Major OpenOffice.org Deployments.* (2009, March). Retrieved from http://wiki.services.openoffice.org/wiki/Major_OpenOffice.org_Deployments.

[16] Rothwell, Richard. (2008, August 5). *Creating Wealth with Free Software*. Retrieved from http://www.freesoftwaremagazine.com/community_posts/creating_wealth_free_software.

[17] *The Open Source Definition*. (2007, July). Retrieved from the Open Source Initiative Web site: http://opensource.org/docs/osd.

[18] Perens, Bruce. (2006, February). *The Emerging Economic Paradigm of Open Source*. Retrieved from http://perens.com/Articles/Economic.html.

[19] Raymond, Eric. (2000). *The Cathedral and the Bazaar*. Retrieved from http://www.catb.org/ esr/writings/cathedral-bazaar/cathedral-bazaar/.

[20] Rossi, Maria. (2004, April). "Decoding the 'Free/Open Source (F/OSS) Software Puzzle' a survey of theoretical and empirical contributions." Retrieved from Universita degli Studi di Siena Web site: www.econ-pol.unisi.it/quaderni/424.pdf.

[21] Serpo, Alex. (2008, October). *Red Hat: Economic crisis to boost open source*. Retrieved from http://news.cnet.com/8301-1001_3-10067617-92.html.

[22] UNU-MERIT. (2006, November). *Economic impact of FLOSS on innovation and competitiveness of the EU ICT sector*. Retrieved from the European Commision's Web site: http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf.

[23] Weber, Steven. (2000, June) The Political Economy of Open Source Software. *BRIE Working Paper 140*. University of California, Berkeley. Retrieved from http://repositories.cdlib.org/brie/BRIEWP140.