If it’s Not Broken, Improve It: Theoretical Expansion of Free, Open Source Software Provision Through Nonprofit Designation

Greg Gorence
December 11, 2012

Senior thesis submitted in partial fulfillment
of the requirements for a
Bachelor of Arts degree in Economics
at the University of Puget Sound

Whether you're a technophile or technophobe, or think we're all really living in the Matrix, computers are an inescapable aspect of life in the present. More than the oversized calculators of yesteryear, modern computers allow the user to archive, access, and create anything that can be represented as a string of 1s and 0s. This can be music, imagery, literature, video… the list could be limitless. Modern computers are also the gateway to the Internet, the cornucopia of all that has been archived, accessed, and created by computer users as a whole. Whether you're accessing the combined knowledge of all experts in a given field or your great aunt is chain-mailing you pictures of adorable kittens that will all suffer unless you forward those pictures to everyone in your contacts, the Internet is a virtually instant and highly scalable platform for global human interaction.

Computers and the Internet have an especially measurable impact where that human interaction is a transaction. Online vendors use the Internet to cut costs associated with brick-and-mortar establishments and to save advertising effort. Consumers can save time shopping online that would otherwise have been spent screening for trustworthiness, price, and quality. Yet, the individual's price of accessing the Internet as a means to such savings could have been profoundly higher if proprietary software had been chosen to structure the Internet instead of Free, Open-Source software (FOSS).

Most e-mail, wanted and unwanted, arrives in your inbox by way of Sendmail; Apache runs about 70% of all web servers; BIND translates the URLs web surfers type in their browser bar (Apple.com, for example) into the IP addresses actually understood by computers and the web server on the other end, for *all* websites (try searching

17.149.160.49 in a web browser’s address bar) (Krishnamurthy 2003, 47). Each aforementioned program is FOSS; together they’re the silk of the World Wide Web.

Though FOSS is gainfully employed as the framework for the Internet, it has not been equally successful as a direct-to-consumer product. Consumption-type FOSS programs such as operating systems are generally disused by those without knowledge of programming languages or communities. Meanwhile users with this information are more willing to install, maintain, and voluntarily produce FOSS. This work’s contribution will theorize that FOSS’ appeal can be extended to more computer users through the nonprofit organization of FOSS producers.

Computer software is a highly complex good. Its creation requires substantial accumulation of human and physical capital, and as I shall argue, use of social capital in the case of open source software. The wages of skilled, costly laborers must also be paid, or in the case of FOSS, foregone. Despite its valuable inputs and its pricy competition, the prevailing price of FOSS is zero. Given its price, FOSS has the potential to extend an individual user’s utility from income greatly, but it has failed to reach users without a complete understanding of FOSS. This work will theorize that the aggregation of FOSS producers into nonprofit organizations could help increase FOSS output from programmers. Moreover, nonprofit status could theoretically make FOSS a more attractive consumer program to the lay computer user.

To understand the current state of FOSS communities and how best to model them, I will first present a review of the past studies. The reader will then scrutinize my model, a theoretical representation of the forces ripening FOSS for nonprofit organization. Subsequently, I will present an analysis of the strengths and weaknesses of this work and suggest some areas that warrant further exploration.

Literature Review

Since the term became ubiquitous in 1998, studies of open source have evolved from intellectual curiosity towards scholarly articles appearing in professional and academic journals. Open source work is freely distributed and redistributed, is open to modification, and is neither exclusive nor exhaustible (opensource.org). Typically, these characteristics are not conducive to the efficient and innovative production of a good.

Andrea Bonaccorsi, Silvia Gianngeli, and Cristina Rossi (2006) recognized that switching costs exist where the user with an established hardware-to-software rapport switches to a new program. Using ordinary least squares regression analysis, they concluded that network externalities are a significant switching cost (Bonaccorsi et al 2006, 1094). These positive network effects manifest wherever the user's benefit from a product increases as that product's user base grows (Cheng et al 2011, 203). Learning curves are another form of switching cost: the FOSS user must solicit the assistance of another user to accomplish otherwise price-provided automatic maintenance and security tasks (Bonaccorsi et al 2006, 1086). Karim Lakhani and Eric von Hippel (2003) tested the efficacy of FOSS help and maintenance through a case study of the Apache Usenet, and concluded that help seekers saved a mean 103.5 minutes of time at the expense of 4 to 9.3 minutes of a respondent's time (Lakhani et al, 933-935). Three years earlier, Jennifer Kuan formed a similar hypothesis, but found that web server bugs were fixed at about the same rate by the free solution and its proprietary counterpart (22). Combining the conclusions of these two studies, my model will treat the amounts of time saved by FOSS solutions and proprietary substitutes as positive and proximate.

Assuming the user has made these choices and opted for the FOSS switch, and is now an integrated user, she faces a new choice: she will either reciprocate or free ride. Should she reciprocate, she will volunteer time towards improving the use experience of all, whether or not the beneficiaries include free riders. Jennifer Kuan’s (2000) study of FOSS controlled for user programming fluency and user willingness to contribute. User willingness to contribute to FOSS is assumed to be equal to user willingness to pay for the proprietary substitute (13). Her empirical analysis concluded that user-nonprogrammers with low willingness to pay or contribute will always free ride with the nonrival FOSS product. Though free riders enlarge the user base, their unwillingness or inability to give back may deter other users from volunteering (Kuan, 12). Users with programming proficiency will reciprocate to a degree reconcilable with their willingness to pay, and user-nonprogrammers with a high willingness to pay will buy the proprietary substitute as long as its price does not exceed that willingness (Kuan, 12). Based on these conclusions, Kuan recognized that the user is a factor of production when she reciprocates.

Krishnamurthy (2003) recognized that FOSS support communities could also double as peer developers and reviewers (51). He asserts that these large, dedicated peer groups create dependably bug-free software. Moreover, each user of the software will benefit from this dependability at no private cost. Opportunity cost of foregone rent will be borne by the pioneer programmer who has to openly share her innovative source code if she wants her product to benefit from peer scrutiny (Harhoff et al 2003, 7). However, she also saves the time she would have otherwise had to dedicate to bug fixing on her own (Krishnamurthy, 51).

Josh Lerner and Jean Tirole (2002) categorized the programmer’s intangible benefits as “immediate payoffs” and “delayed payoffs (213)”. Immediate payoffs, they assert, accrue to the contributing programmer as she is writing the software or immediately afterwards. Such benefits may include, but will not be limited to: the improvement of professional skill, which begets both immediate and delayed payoffs, and: the chance to work on a “cool” project that could yield game-changing, breakthrough innovations (Lerner et al 2002, 213). Delayed payoffs stem largely from those immediate payoffs already mentioned. The honing of professional skill will likely translate into opportunities for the programmer to boost her pay grade or become more competitive in the job market. Likewise, being the co-creator of a breakthrough innovation will earn the programmer respect and recognition amongst his or her peers (Lerner et al, 213). This kind of peer recognition is easily understood as a form of social capital that will facilitate cooperative efforts for the programmer in the future. Using ordinary least squares regression, Crowston and Scozzi found evidence that FOSS projects directed by prominent leads were significantly more likely to succeed (Crowston et al, 13).

Lerner and Tirole found these payoffs when they sought to answer the question: “Why should thousands of top-notch programmers contribute freely to the provision of a public good?” (198) They asked because the costs of such an undertaking are considerable. Programmers are skilled, creative people who are constantly expected to sharpen the cutting edge, and that makes their time valuable. Plush salaries make for a high foregone wage (Lerner et al, 213). They also argue that the programmer’s employer will suffer time theft if she’s not constantly engaged with the corporate paradigm while on the clock (213).

While most open source software is conditionally free, it is not by the conditions of its definition or license. Free open source is free only by the choice of its author, who has decided that her work will be distributed at a zero price. In a working paper from 2003, Dietmar Harhoff, Joachim Henkel, and Eric von Hippel asked why the apparently altruistic FOSS originator shares their valuable intellectual property freely. They found their answer in a number of benefits that accrue to the programmer who voluntarily shares valuable product and process innovations. The authors concluded that innovation sharing helped FOSS communities overcome transaction costs, informational asymmetries, and contract monitoring and enforcing costs (Harhoff et al, 20). Proprietary ownership contracts are increasingly difficult to monitor as a result of online piracy, meaning contractual efficacy must be bought at increasing cost; bypassing this cost is a benefit arising from the FOSS choice. It is also costly to the programmer to improve and maintain the performance and security of their software. The authors recognized that these costs could be borne by others if the programmer freely shares source code (11). Most importantly, the authors concluded that involuntary information spillovers under a proprietary license were a cost, while voluntary information spillovers resulting from the free distribution and modification of open source programs were a benefit (20).

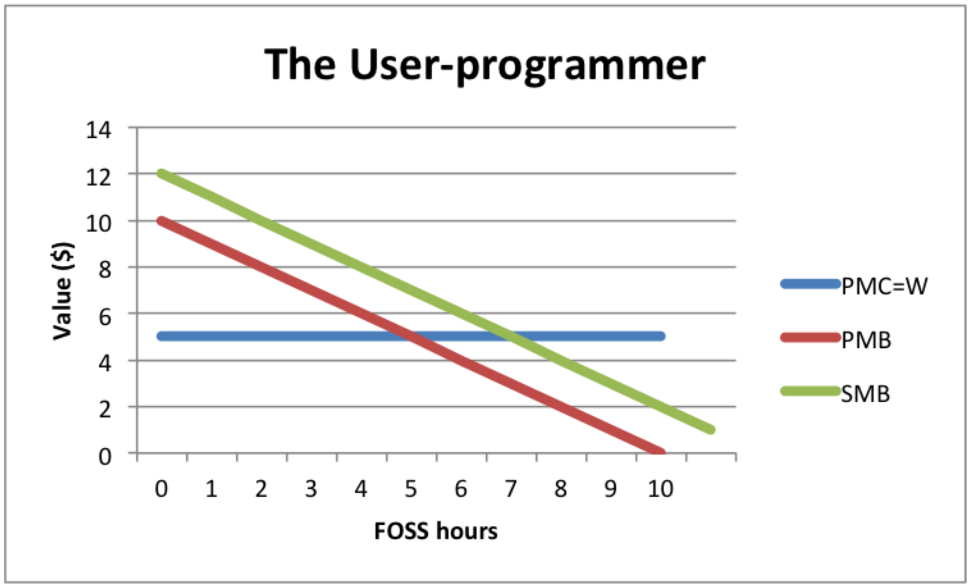
Modeling the Effects of Nonprofit Designation on FOSS Economies

Software is a good that provides valuable services to a user through its consumption. The employment of software enables the private user to more easily and affordably store, share, and scale information. This could be family photos; it could also be business ledgers. The bottom line is that information can be stored in bytes instead of on bookshelves, and hard disk space is much cheaper than shelf space. However, any for-profit producer of software would quickly go out of business if they asked nothing in return for their product. After all, software production requires the rent of technologically advanced capital and the employment of highly skilled labor. In a typical market for such a complex good, consumers will buy so long as the next consumer's expected utility from the good is at least equal to the price they pay, and producers will supply the good to a point that the cost of providing the next unit does not exceed the revenue it expects in return. Once a software program is completed, the cost of distributing it from the first to the last consumer is negligible. For-profit producers sell their output to recoup their expenditures on fixed costs.

What sets the software market apart from other markets that provide a comparably complex good is the participation of producers that give their product away to users. Users are excluded from the consumption of FOSS only if they lack the means to obtain it. Considering that almost 80% of the US population is connected to the Internet through a home personal computer, those excluded are few (Google.com). Nor is the amount of FOSS currently available on the market diminished by those who have consumed it before. The software market is unique in that a market-allocated complex private good shares that same market with an equally complex quasi-public good.

Free-riding users of FOSS enlarge its user base but cannot expand the pool of labor and capital that the project can rely on. A larger user base could attract more user-programmers trying to signal employers. Conversely, free riders cannot or will not give back, which may deter user-programmers looking for peer review. These counteractions will affect a user-programmer's private marginal benefit curve, depending on what that programmer wants in return from the community.

User-programmers volunteer their skilled labor and high-rent capital towards maintaining and advancing FOSS projects. The user-programmer has invested considerably in her training and equipment, and will always possess the factors needed to produce FOSS. If the user-programmer is jobless or in training, she can send signals to employers through FOSS production. Otherwise, the opportunity cost she bears to volunteer the next hour of FOSS production is assumed equal to her wage. Her voluntary product will in turn be used by individuals, professionals, students and other members of society who somehow benefit. However, as is the case with other public goods, the private provider meets only her needs, leaving society's unmet.



(1)

Using the preceding chart (1) as a reference, society's needs for functionality and quality could be met if each private user-programmer volunteered two more hours. She won't do this unless motivated further. If the private marginal benefit she derives from volunteering the next hour can be increased by two per unit hour, her private marginal benefit will coincide with the marginal benefit that society enjoys from her time. Formalizing productive communities of user-programmers into nonprofit organizations could help motivate them to volunteer further. It may also motivate inactive users with programming fluency to begin volunteering. In both cases, the end result would be more hours volunteered.

Large and established FOSS projects have gained 501c(3) status in the past. Nonprofit status has allowed the Apache Foundation to accept tax-deductible donations from the public it serves. Evidently, nonprofit status has motivated private donors to support Apache’s mission: donations have grown $520,608 since Apache’s 501c(3) designation in 2001. In 2011, Apache’s revenue outweighed its expenses by $133,928 (Apache.org 2011). According to Apache’s form 990, none of that surplus went towards the compensation of volunteers for their product-enhancing contribution (Apache.org). Nor did it give grants to smaller FOSS projects. Large FOSS producers are more visible to donors of time and money. If that visibility begets more revenue than is necessary for operating expense, the remainder could be used to compensate user-programmers for their contribution, effectively increasing their marginal benefit.

There should be a limit to what kind of contribution is considered valuable. A contribution that serves the reliability and broad functionality of the program ought to be eligible for remuneration. Those additive contributions that serve a narrow niche user’s utility are best left unsubsidized; they would still occur independent from remunerative motivation (Oreg et al 2007, 2058). Likewise, undesignated FOSS producers whose product cannot reach a broad user base would not deserve or need for donative funding.

Conversely, a startup program that could potentially advance the way society uses computers deserves the support of motivated donors and programmers. Donors could be private users or established FOSS foundations. Their contribution would help motivate user-programmers to volunteer more time towards a breakthrough innovation.

Monetary recompense aside, nonprofit status could motivate the volunteer user-programmer in more subtle ways. Nonprofit designation sends a trustworthy signal (Hansmann 1980). Computer users choosing between a FOSS program and its proprietary substitute may be more inclined to choose the former if it's produced by a designated nonprofit organization. Furthermore, users who appreciate the nonprofit signal will be more inclined to donate if they know their donation serves the quality and functionality of the product they use. User-programmers might be more willing to volunteer for a nonprofit due to several reasons. A nonprofit FOSS producer would be more visible than its undesignated counterpart, and would draw more attention from corporate recruiters, community members, and donors. A significant contribution to a visible product could thus lead to a better job and income. It could also lead to elevated status amongst one's peers and a windfall of social capital. But most importantly, a significant contribution improves the quality and functionality of a good that is publicly available free of charge. That improvement will attract more users and donors.

Social capital economies within the FOSS community deserve special consideration. Frequently, user-programmers within the community are employed at for-profit software companies. Volunteering FOSS hours serves as a way for the employed programmer to stay engaged and hone their programming skill. If they're happy with their employment, they may not have much left to gain from the FOSS community.

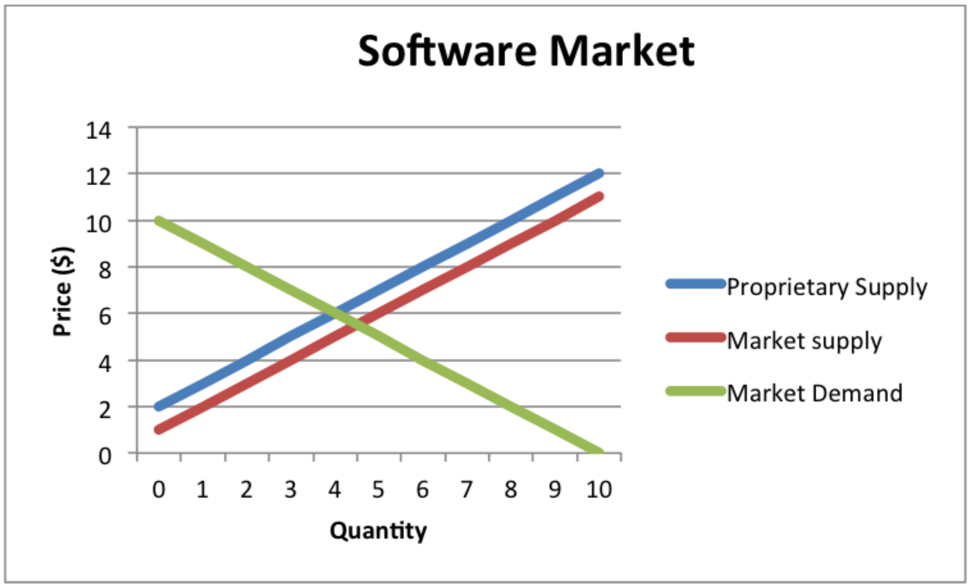
However they have much to give. As a newly trained or currently enrolled programmer, catching a FOSS veteran's eye could create job opportunities. To receive that kind of attention, the newcomer's contribution would have to be significant and visible. The newcomer's visible success might also lead to stronger and broader community ties down the road; those ties could be used to form a productive FOSS cooperative at a lower risk. Without confidence in a strong lead, it can be difficult for a FOSS cooperative's vision and productivity to remain undivided (Crowston et al, 15). Failure to unite a cooperative's productive forces towards one goal could mean the failure of the project itself.

Social capital is needed to strengthen the bonds of business and trust in the capitalist economy where reward requires risk (Salamon 1999, 17). The individual's risk can be reduced if she pools it with that of others. If she knows and trusts her partners, that risk will be further decreased (Valentinov 2004). Nonprofit status could help expand the risk pool through the inclusion of donors. It could also help deepen the risk pool through the availability of risk-abating information to volunteer programmers and donors; made possible by the publication of the Return of Organization Exempt from Income Tax, otherwise known as Form 990. Through the enlargement of the risk pool and the reduction of private risk, the volunteer user-programmer would more readily spend her time on FOSS production.

Though FOSS is distributed free of charge, it is a good that society can benefit from tremendously. It is a tool of trade, having lowered the cost of communication and transaction. It is also a practice space for aspiring programmers who will someday deliver the next leap forward in computing. Most importantly, FOSS offers a legal good to low-income users who might otherwise pirate a proprietary product. Historically, one of the roles of government has been to provide socially beneficial public goods and services. Extending the nonprofit designation to productive and innovative FOSS cooperatives would expand the government's ability to finance breakthrough FOSS innovations with positive social implications.

Government sponsorship would have its greatest effect during a FOSS initiative's startup phase. Startup costs are otherwise borne by the initiative's user-programmers (West et al 2005, 2). More FOSS initiatives could be started and completed if government grants footed some of the startup costs and reduced the riskiness of startup to the individual and her partners.

FOSS is a product distributed free of charge. Its proprietary substitute can either be purchased or pirated. To low-income consumers, the choice is reduced to either FOSS use or software piracy. The improvement of FOSS quality and functionality through the motivation of volunteer user-programmers would make it a more appealing alternative to piracy. Moreover, if nonprofit designation served to signal FOSS' quality gains in relation to the proprietary alternative, the proprietary competitor would have to either lower prices or otherwise distinguish itself. Illustrated in chart (2), either outcome would positively affect the benefit that society enjoys from the software it uses. Nonprofit designation and government subsidy might also popularize FOSS amongst other public-serving providers of goods and services. Choosing volunteer user-programmers to assume the provision of FOSS programs and services would reduce software costs to public utilities and public educators.

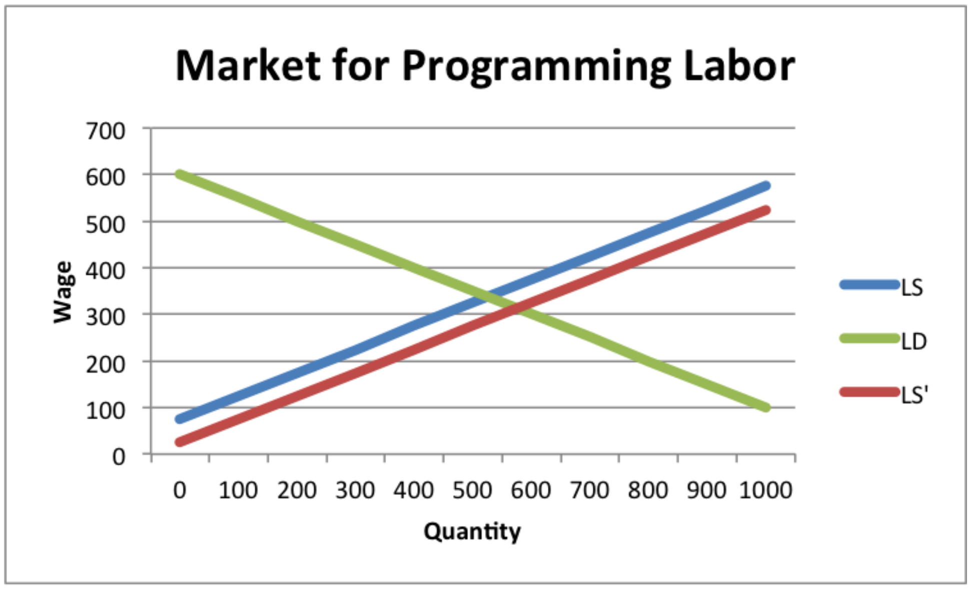


(2)

Private users of software enjoy increases in positive network externalities as the user base of their program grows. User base expansion occurs either when more users choose a single program or when a substitute program gains cross-compatibility with the program of interest. It is in the FOSS producer's best interest to support cross-compatibility. Thereby delivering a larger user base and increased positive network externalities to the FOSS program's users. Proprietary producers will opt for a different strategy: the prevention of cross-compatibility (Cheng et al, 233). If a computer user wants to enjoy the network externalities of a non-compatible proprietary product, she must either buy it or pirate it.

For the argument at hand, we will assume this user's utility is most served by her perceived network externality gains. Additionally, she has no desire or ability to program. She will perceive the proprietary program's out-of-box network size, quality, and reliability as superior to that of the FOSS substitute. Her perceptions and ultimate choice will change if FOSS' network size and quality can be more effectively cultivated and broadcast. Through the motivational mechanisms caused by nonprofit designation outlined above, the actual and perceived user base will expand. FOSS' programming base will also expand its actual quality, as volunteers are motivated to contribute fixes and functions. Nonprofit designation will also serve to make FOSS a product of perceptibly higher quality. Moreover, if the FOSS program achieves cross-compatibility its users will enjoy the cross-network externalities of substitutes. Delivering cross-compatibility can be a challenge to FOSS producers, but it can be more easily delivered by nonprofit producers that have access to a larger and more motivated base of user-programmer volunteers.

To more truthfully measure the consequence of nonprofit designation on software economies, its effects on the programming labor market warrant further investigation. Its effects on the private user-programmer were hypothesized to result in an increased willingness to contribute. User-programmers who volunteered before nonprofit designation would volunteer more; users with programming fluency who hadn't contributed beforehand would more likely do so. The result visible in chart (3) is an increase in the labor supply to FOSS producers. This reduces the amount of hours the private user-programmer must volunteer without reducing the output or quality of the FOSS organization as a whole. The positivity of this outcome could be threatened by a counterforce that would also be induced by FOSS nonprofit designation.



(3)

The extension of donations and government funding to startup FOSS initiatives would increase those initiatives' probabilities of successfully reaching maturity. An increase in the amount of mature FOSS programs would mean more demanders of volunteer programming labor. If consequently the demand for labor increased faster than the supply, an immediate labor shortage would stall program quality and functionality growth past maturity. Failing short-term labor supply expansion, the result would be a deadweight over-allocation of public funding; imposing a cost on society as opposed to extending a benefit.

User-programmers who volunteer their time towards FOSS production cannot rely on a reward to make a living. They must be otherwise gainfully employed, typically at a for-profit software firm. If nonprofit designation successfully fostered FOSS program quality and functionality growth in relation to its for-profit competitors, those for-profit producers would have to lower the price of their good to remain competitive. A few firms may even shut down. Subsequently the amount of demanders that are willing to pay for programming labor would fall. Fewer positions would be made available, and fewer programmers would be able to successfully find work in the industry they trained for. Those user-programmers who contribute to FOSS growth may actually be figuratively shooting themselves in the foot by shrinking the demand for the labor they are trying to sell.

Critical Analysis of the Model

The model presented in this work implicitly assumes that FOSS is a substitute to proprietary software. Though the lion's share of literature reviewed agrees with this assumption, James Bessen's 2005 work asserted that this assumption is untrue (Bessen, 4). He argued that users of computer software are dissimilar in their requirements for software functionality. The lay user only needs basic tools such as web browsers and word processors. Proprietary software dependably provides uniform, yet basic functionality. Advanced users that ply their trade by way of computers and networks share that need for the basics, but require savvier tools for their trade. Bessen theorized that FOSS' versatility and malleability delivered those advanced users a means to make their own tools. That it was in fact complementary to proprietary software for this reason (Bessen, 4).

Bessen's model holds true for FOSS programs that serve a narrow user base. In line with his theory, the production of professional-grade FOSS programs would not benefit greatly from nonprofit designation. This type of program will be created autonomously by the private programmer for the private programmer's benefit. As such, it neither needs nor deserves public support. The model presented attempted to compensate for this through limitation of the type of FOSS initiative that deserves public support. Those FOSS programs that will potentially deliver benefits to all of society's users generally deliver small private benefit to the private programmer that produces it. Therefore, the socially beneficial FOSS program will be underproduced by the unregulated market and is deserving of the public support that can by delivered by nonprofit designation.

Producers of FOSS are disallowed from rent collection by way of its very definition (Opensource.org). Designating those producers as nonprofit organizations may simply be a redundancy that introduces nothing new to the economy. The FOSS definition’s prohibition of rent collection has served to strengthen the bonds of trust amongst all user-programmers within the FOSS community. The private programmer can contribute to the community knowing that no single community member will use her contribution for private gain (Harhoff et al, 5). This model has nominated nonprofit designation as a means to incorporate users outside of the programming community into its bonds of trust. Moreover, nonprofit designation would help the FOSS community more powerfully express the qualities that it already possesses. Thus attracting more programmers and users who value quality, reliability and functionality.

The FOSS community has heretofore picked its own winners and losers. Programmers are chosen on their merits, and not by their ability to attract donors. Project leads enjoy no formal authority, but are endowed instead with the trust and willingness of others who have worked with or observed the leader in the past. Those projects spearheaded by a well-endowed leader are more likely to succeed and reach maturity (Crowston et al 2002, 4). This would be a perfect mechanism for allocating the FOSS community’s resources if the community and its leaders always picked projects based on their social merits. However, the private volunteer programmer picks whom to follow and what to contribute based on personal preference. That personal preference hampers the provision of socially beneficial FOSS. The private user-programmer would enjoy no private benefit from using the water management or elementary math program she helped build, but society would benefit greatly from her contribution.

Presently, proprietary software producers dominate the provision of programs that are broadly used and enjoyed in society. Commonplace programs such as productivity suites (e.g. Microsoft Office) have the most potential social benefit due to their widespread use, but the producer instead captures those benefits as rents. This market coverage is strengthened by the size of the proprietary program’s user base, which increases private positive network externalities as it grows (Cheng et al, 223). Larger perceived network externalities in turn make it a more attractive good to new users. If market coverage is made too strong or complete, it will prevent the FOSS producer from entering the market competitively. Users will elect to stay with the proprietary program, as switching to the FOSS substitute would incur switching costs. Moreover, the user will lose the positive network externalities they enjoyed while using the proprietary program. The user’s positive externalities will only be preserved if the FOSS producer can create lasting cross-compatibility between its program and the proprietary program. However, as Cheng notes, the proprietary producer will always rebuff the FOSS producer’s efforts to support cross-compatibility (223). After decades of operating in the niche, FOSS may be too small relative to proprietary producers that have successfully covered the market for broadly used programs.

Mozilla’s Firefox web browser lends credence to the theory that nonprofit designation can help a FOSS program compete in an otherwise covered market. Its speed and reliability has attracted users away from proprietary substitutes such as Internet Explorer and Safari. However, Firefox is more the exception than the rule. Its success is also attributable to its heritage as FOSS that was created by a for-profit software producer. Considering its heritage allowed it steady access to capital and labor in the startup phase, it cannot be a parallel case to FOSS startups that must rely on volunteer labor and decentralized capital to succeed.

The nonprofit designation's ability to sway consumer preference and perception is also debatable. Moreover, its efficacy will vary with the industry of interest. Lay users of computer software tend to appreciate standardization, not heterogeneity. Proprietary producers can more readily provide those appreciable regulations and standards. Given that FOSS producers provide no user's manual or documentation with their product, nonprofit designation may do little to improve its perception in the eyes of the lay user who prefers a standardized product.

Concluding Remarks

Proprietary software producers currently dominate the market for broadly used end-user programs. Those programs can be used and enjoyed by anyone, not exclusively by programmers and professionals. Due to the size of their user bases, such programs could deliver the greatest social benefits. However, proprietary producers use their market position to capture those social benefits as rent. Moreover, they use that rent to enforce the contracts that protect their private intellectual property. Free, open source software could be a means for society to capture those social benefits. Herein lies a catch: the unregulated FOSS market underprovides socially beneficial programs. Private user-programmers volunteer their time for the program that best serves their private benefits, and will stop volunteering when it is no longer worth their time.

Proprietary market coverage has also tied many users into the proprietary scheme. Users who transfer to the FOSS program incur the need to learn a new program's nuances. They also forego the standardization and large user bases that proprietary programs provide. Those sacrifices make FOSS much less attractive to the user that already knows and employs the proprietary program. Though FOSS is equal in quality and functionality to proprietary software, it suffers the lay user's misperception of inferiority.

Society could benefit greatly from the motivation of volunteer user-programmers to increase the production of broadly used and hence socially beneficial FOSS programs. Those benefits would only be realized if lay users could be convinced to make the switch to FOSS. As users switched, the network would grow and in turn attract more users seeking positive network externalities. In order to initiate this "positive spiral" of user growth, the perceptions and preferences of computer users must be changed in favor of FOSS.

Nonprofit designation is one way that firms in other industries have swayed consumer opinion. Nonprofit designation forms a bond of trust between the producer and consumer where profiteers cannot. Trust lowers transaction and information costs, and aids the formation of low-risk trade. Those bonds of trust already exist amongst the active participants in the FOSS community. Nonprofit designation would be a means to extend those bonds to the end-user who cannot participate. An end-user's knowledge that they received a useful good of high quality will prompt her to give back to the FOSS community; nonprofit status serves as a means to collect valuable donations from the appreciative user and from society as a whole. That end-user's indirect participation in the production of FOSS and direct participation in FOSS' networks will attract more volunteer user-programmers and lay users, initiating a positive spiral of user growth akin to the one that swept Microsoft to the top.

Bibliography

1. Apache. "Fiscal year 2010-2011 Annual Report." April 30, 2011. http://apache.org/foundation/records/990-2010.pdf

2. Berdou, Evangelia. *Organization in Open Source Communities: At the Crossroads of the Gift and Market Economies*. 1st ed. Hoboken: Routledge, 2010.

3. Bessen, James. "Open Source Software: Free Provision of Complex Public Goods." *Boston University School of Law and Research on Innovation* (2005).

4. Bonaccorsi, Andrea and Cristina Rossi. "Comparing Motivations of Individual Programmers and Firms to Take Part in the Open Source Movement, from Community to Business." *Laboratory of Economics and Management* (2006).

5. Bonaccorsi, Andrea, Silvia Giannangeli, and Cristina Rossi. "Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry." *Management Science* 52, no. 7, Open Source Software (July 2006): 1085-1098.

6. Casadesus-Masanell, Ramon and Pankaj Ghemawat. "Dynamic Mixed Duopoly: A Model Motivated by Linux Vs. Windows." *Management Science* 52, no. 7 (July 2006): 1072-1084.

7. Cheng, Hsing Kenneth, Yipeng Liu, and Qian Tang. "The Impact of Network Externalities on the Competition between Open Source and Proprietary Software." *Journal of Management Information Systems* 27, no. 4 (Spring 2011): 201-230.

8. Crowston, Kevin and B. Scozzi. "Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development." *IEE Proceedings* 149, no. 1 (2002): 3-17.

9. Crowston, Kevin, Kanging Wei, James Howison, and Andrea Wiggins. "Free/Libre Open-Source Software Development: What we Know and what we do not Know." *ACM Computing Surveys* 44, no. 2 (April 2012): 7-35.

10. Dahlander, Linus and Mats Magnusson. "How do Firms make use of Open Source Communities?" *Long Range Planning* 41 (2008): 629-649.

11. Google. "Google Public Data: Internet Users as Percentage of Population." Last modified October 31, 2012. https://www.google.com/publicdata/explore?ds=d5bncppjof8f9_&met_y=it_net_user_p2&idim=country:USA&dl=en&hl=en&q=internet%20users%20us

12. Hansmann, Henry B. "The Role of Nonprofit Enterprise." *The Yale Law Journal* 89, no. 5 (1980): 835-901.

13. Harhoff, Dietmar, Joachim Henkel and Eric von Hippel. "Profiting from Voluntary Information Spillovers" How Users Benefit by Freely Revealing their Innovations." *MIT Sloan School of Management* Working Paper 4749-09 (January 2003).

14. Kuan, Jennifer. "Open Source Software as Consumer Integration into Production." *Stanford Institute for Economic Policy Research* (2000).

15. Krishnamurthy, Sandeep. "A Managerial Overview of Open Source Software." *Business Solutions* (September 2003).

16. Lakhani, Karim R. and Eric Von Hippel. "How Open Source Software Works: "Free" User-to-User Assistance." *Research Policy* 32 (2003): 923-943.

17. Lerner, Josh and Jean Tirole. "Some Simple Economics of Open Source." *The Journal of Industrial Economics* 50, no. 2 (June 2002): 197-234.

18. Lerner, Josh and Jean Tirole. "The Economics of Technology Sharing: Open Source and Beyond." *National Bureau of Economic Research* Working Paper 10956 (2004).

19. Oreg, Shaul and Oded Nov. "Exploring Motivations for Contributing to Open Source Initiatives: The Roles of Contribution Context and Personal Values." *Computers in Human Behavior* 24 (2008): 2055-2073.

20. Salamon, Lester M. *America's Nonprofit Sector: A Primer.* 2nd Ed. United States: The Foundation Center, 1999.

21. Valentinov, Vladislav. "Toward a Social Capital Theory of Cooperative Organization." *Journal of Cooperative Studies* 37, no. 3 (2004): 5-20.

22. West, Joel and Siobhán O'Mahony. "Contrasting Community Building in Sponsored and Community Founded Open Source Projects." *San Jose State University Faculty*

Publications Proceedings of the 38th Hawaii International Conference on System Sciences, (2005).