The Economics of Open Source Software for a Competitive Firm

Why give it away for free?

Richard E. Hawkins (dochawk@scualum.com)
Pennsylvania State University, Dubois

November 12, 2002

Abstract. Large quantities of software, ranging from operating systems to web servers to games, are now available as "open source software" or "free software." In many cases, this software is backed by large profit seeking corporations such as IBM. Traditional economic analysis is used to identify the costs and benefits to firms of using open source rather than proprietary solutions, particularly in the case of the firm releasing code to the world when not obliged to do so. Examples of large companies backing open source are examined in light of the profit motive. Additionally, open source is also analyzed as a quasi-public good.

Keywords: free software, open source software, public good, competitive firm

1. Introduction

In the early days of the computer industry, software was not seen as a product. Instead, it was something that was given away to help sell hardware. This model continues for some firms to this day, such as with Apple Computer, which considers its business to be hardware, but develops its own operating system to distribute with the hardware. Apple deviates from the older model in that it does charge for updates of this software.

Today, rather than being a means to sell hardware, software is profitable in its own right. Microsoft enjoys gross profit margins in excess of 89%, (Wall Street Journal, 2002)–a level rarely seen in any industry. Microsoft's royalties from the inclusion of Windows and Office with a computer frequently exceed the net profit for the seller of the hardware. Businesses spend billions of dollars annually on software. A new trend has developed in recent years, that of "free software" or "open source software," or "OSS," in which the software is distributed without charge to the recipient, and with the source code, such that the the code may be modified and redistributed.

This trend has resulted in (among others) the Apache web server, which has approximately two thirds of the market and two and a half times the market share of Microsoft (netcraft.com, 2002)–its closest commercial rival; Linux and the BSD family of operating systems, which tests have found to be more stable than Windows and at least

comparable in performance; and perhaps most interesting to economists, projects such as Apple's Darwin project (developer.apple.com.darwin) and Netscape's Mozilla project (mozilla.org), in which firms have turned (or attempted to turn) proprietary software into open source software, presumably expecting greater profits than if the code were kept proprietary; and Corel's involvement in the WINE project (Linux Today, 1999)– apparently finding that its commercial interests were aligned closely enough with Wine's agenda to be worth Corel's while to finance engineering work on the project, even though Corel's competitors could benefit from the code created. Also of interest are the decisions of firms who abandon or set aside existing proprietary software and instead contribute to and distribute free software, such as IBM with the Apache web server (Wall Street Journal, 1998), and Silicon Graphic's decision to use Linux rather than its own existing Irix for portions of its hardware line (Wall Street Journal, 1999).

Investigation will turn first to the simpler question of adoption of existing open source software for internal use, and then to the economic questions faced by a firm considering an existing or new open source project as a source of revenue.

2. Existing Literature

While this is not the first paper to address economic issues in open source software, it is the first to focus on the choice of a profit-seeking firm to release its product in such a manner. The existing literature can primarily be grouped into advocacy works, explanations of the culture of the programmers, those that focus on the production process, and those that consider the release of open source software as a strategic move. The latter material is closest in nature to the model herein.

Several authors make no apparent distinction between the "General Public License" of the Free Software Foundation (fsf.org) and free software in general. The GPL is proceeded by a "manifesto," setting forth the ideological agenda, which must be included with the license itself. It also uses the term "copyleft" to describe itself. Even though the term "copyleft" is deliberately loaded with ideology by its creators, the analysis of Mustonen (Mustonen, discussion paper) considers no other possibility when considering it as a competitor to a monopoly product. Perhaps the best known paper relating to the subject, (Lerner and Tirole, 2002), notes the "viral" nature of the GPL discussed below in section 4.1.2, and recognizes a decline in the usage of the GPL in favor of other licenses–while apparently mistaking Debian's "Open Source Definition" (Open Source Definition) and "Social Contract" (Debian

Social Contract) for licenses rather than policy positions by the Debian organization (which has close ties to the FSF).

A few of the papers focus on the culture or motivations of programmers who work in their free time to produce free software. Lerner, (Lerner and Tirole, 2002), for example, considers altruism and status among other programmers and users as a motivation for the individual contributor. The status is valuable not only in its own right, but the reputation gained may lead to a higher-paying regular job. Johnson (Johnson, working paper) considers production with a population of user-developers, who estimate the chance of needed software appearing and either develop or not. While this analysis is consistent with the special case of development or modification for in house software considered below, it does not cover the general case of the profit-seeking firm.

Other papers consider the open source process either as an engineering management issue or for the effect on industry structure. Mockus (Mockus, 2000) considers the engineer aspects of development, and traces development of the Apache Web server and the handling of organizational issues for the project. While IBM and other commercial entities provided engineers, the focus is on the process rather than the motivation of the firms. von Hippel (von Hippel, working paper) addresses the organizational science issues presented by open source development, while (Khalak, working paper) focuses on adoption of open source by commercial users of software, and the potential impact upon commercial software vender.

The final economic papers of interest are those that consider open source as a strategic behavior by firms. Pal(Nilendu and Madanmohan, working paper) considers the possibility of a software solution split into two portions, one of which may be open source, and the other kept proprietary. The commercial motivations considered in the paper are through sale of support, such as with Red Hat, and from sales of the proprietary portion of the solution. Lerner (Lerner and Tirole, 2002) also considers a strategic motivation for releasing open source–the weakening of a competitor or the other member of a duopoly.

In addition to the papers directly addressing open source software, there are also papers considering other industries in which intellectual property is released without charge–a situation comparable to releasing open source code. Lecocq (Lecocq), on the other hand, considers the structure of the role-playing game industry in the United States, in which a firm designs a game and other firms make complementary products for use with the game. In a similar light, Harhoff (Harhoff, et al, working paper) considers gains to firms in some industries for revealing technologies, such as chip construction techniques developed

by IBM and revealed to other producers. In these cases, the innovator profits from the larger market created by revealing portions of the technology, either by sales of the base unit or by selling equipment to other producers.

3. Model for a Software Consumer

When a firm undertakes a project, it is with the assumption that the value to the firm exceeds the cost. The purchase of a computing system is no different in this regard from any other input. For a computing project, the costs can be divided into hardware costs for the purchase of equipment E, the purchase of the software at a price B, internal administrative costs A, external support costs S, and down time costs D to the firm of down time when the system is unavailable due to failure or while waiting for repair of hardware or software, for a total cost C.

To make such an investment, the value to the firm of the project must meet or exceed this cost. Aside from highly specialized projects (for which the existence of an open source solution is unlikely), the purchase price of software is typically a small portion of the total cost of a computer–even at $1,000, it is less than the cost of an employee for two weeks or the annual support costs for the machine upon which it runs. While anecdotal claims are made that OSS generally requires less computing resources than proprietary solutions (in particular, than Windows and related software), there seems to be no actual comparison available,[1] and it will be assumed that E is the same for both systems.

The differences will then come from A, S , and D and will be largely an empirical question.

Vendors of proprietary software would naturally like buyers to believe that there is no support available for OSS products. Advocates of OSS naturally claim that adequate support is available on-line, through newsgroups, websites, and mailing lists. The reality is that external support for open source software can be purchased from a wide variety of vendors including IBM and Oracle, while it is available only from the manufacturer for most proprietary software, and any price difference for a given measure of support would seem to come from the monopoly power of the proprietary vendor.

[1] There are comparisons and studies, but all seem to have serious design or methodological problems. Most seem to come from Linux advocates or to have been arranged by Microsoft.

In choosing between existing products with comparable features, the consumer's problem then is simply to

$$\min C = A + S + D \tag{1}$$

But since it is assumed that

$$A \ll C \tag{2}$$

there is no compelling reason to use or not use open source products; the other costs involved drive the decision.

Overlooked in this discussion of the consumer is the possibility that the consumer will customize the software, an oft-touted advantage of open source software. In this case, however, the consumer is a producer, and the appropriate analysis is that of Section 4.

The question remains as to whether the quantity of support needed is higher or lower for OSS. This is fundamentally an empirical question, and no comprehensive studies exist at this time.

4. Model for a Software Producer

Just as it is easy to explain why a firm would pay women less than men for the same work, but impossible to explain why it would pay a man more than a woman, it is far easier to explain why an economically rational agent would use a free software product than why it would give one away. While the questions for a software consumer were primarily empirical, the producer of software also has strategic issues to address. Particularly, the producer must consider the potential benefit to competitors from the the use of the software it develops in addition to any lost revenue.

The model for a software producer will include the firm which uses the software only internally. While such a firm is typically taking an existing program and adapting it for internal use, its advantages and choices regarding redistribution of its changes are the same as for a firm producing for redistribution and have little in common with the simple consumer in Section 3 (Recall that a consumer using the "canned" product is driven primarily by support issues.).

4.1. Assumptions and General Types of Licenses

For the sake of simplicity, it will be assumed that there is no initial cost to the firm to acquire the existing code for the product. Additionally, it is assumed that the firm may redistribute the modified product without

payment of royalties to prior authors and copyright holders. The firm may charge for both support and the purchase of the product. Two types of license will be considered: "public" and "viral."

4.1.1. *Public License*
A "public" license, such as the BSD,[2] X, and MIT licenses, essentially place no restrictions on the use of the code, requiring only a notice of the original copyright in any redistribution in source or binary form. While the code is not actually placed into the public domain, there is no usage on its restriction–it may be used by any person or entity for any purpose. Changes may be made to the source code by a firm that then keeps the changes private and charges for the resulting binaries. As an example, Microsoft uses the BSD TCP/IP stack in various versions of Windows. It has modified the code, but not released the changes. The code may also be changed and re-released under another license, provided that the copyright notice is included.

4.1.2. *Viral Licenses*
Another popular license is the "viral" license, such as the Free Software Foundation's "General Public License,"[3] or GPL. Under such licenses, the source code may be changed, but any distribution of a binary file must make available all changes to the source, and remain under the same license. Furthermore, the recipient of such source may redistribute as often as he wants without any further payment. A firm releasing a product under such a license is forced to accept the fact any person can use the product without payment.

Additionally, viral licenses typically require that all other programs and libraries upon which they depend, save the operating system itself, be distributable under the viral license. That is, to use code from another license, it must be assimilable by the viral license (the "Borg" property). Thus many other licenses are called "GPL incompatible," as they may not be relicensed upon distribution with the GPL code. Conversely, the GPL is incompatible with all other licenses.

4.2. Firm Revenues

"Revenues" will be defined loosely, so as to include both the firm producing for outside distribution and the firm producing for internal

[2] An older version of the BSD license had the additional requirement of acknowledging University of California source in advertising. While the language still appears in some places, the Regents of the University have now waived this requirement under all circumstances.

[3] Also known by its detractors as the "General Public Virus," or GPV.

needs. There are four major potential sources of revenue: sales, support, increased hardware sales, and the value of internal use.

It is the first two, sales and support, that correspond to the direct sale of software. The initial purchase price creates revenue V, while the support contract, if any, creates additional revenue S. A firm drawing from source with a public license and modifying it can then "close" the source and effectively collect V, such as Microsoft's use of the BSD code mentioned above, or the BSDi operating system (which was derived from BSD Unix under the BSD license). Under this scenario, the vendor distributes only the binary with the requirement that it not be further restricted. The selection of an open source base with a public license is then quite understandable–the firm obtains a preliminary version of the product, or a portion thereof, with reduced development costs. It is possible, however, that the firm will still choose to release the source, with or without limitations, for the reasons discussed in Section 4.3.

The firms' options are far more limited, however, with a viral license.[4] While the firm can sell the developed software for a price, it has no ability to stop the purchaser from redistribution of the software. Under such a limitation, it is not realistic to expect sales to be a revenue stream. It is still possible, however, that the firm can sell support for the product and generate revenue. There are classes of software, such as databases, for which support contracts are typically purchased by customers, often at a price far greater than the purchase price. Red Hat and other vendors of software rely on this model: while anyone can download the software from Red Hat's web servers at no price, Red Hat also sells corporate and personal support contracts. While its early profitability came from the sale of boxed software with support for a limited period of time, Red Hat does not expect this to be a plausible long-term revenue source. Instead, it is looking to support fees, including those from OEM's who pay Redhat for assistance in installing Linux on shipped units.

Some firms are fundamentally hardware sellers, but need software to make their machines marketable. Prior to the personal computer, manufacturers typically produced one or more operating systems for their hardware. Apple continues to do so, and many manufactures have shipped slightly different versions of Unix. The variations have forced each manufacturer to maintain and continue development of its own version. Several attempts have been made at unification, but none have met with more than limited success. For this type of manufacturer,

[4] Even if the firm wishes to release its work virally, it may freely dispose of the greater freedoms of the public license.

the software generates no revenue on its own, but allows the sale of additional hardware.

Alternatively, the software may create a demand for the hardware– Sun purchased Star Division, the maker of StarOffice, a Microsoft Office compatible office suite, and continued to give away the program. Sun has now created the OpenOffice project, and released most of the source code under a viral license. Under the terms of the license, OpenOffice may be freely distributed, but Sun has special rights allowing Sun to incorporate additional features into the software without releasing the source, and now charges for the latest version of StarOffice. Sun also provides engineers to the OpenOffice project for development. Sun's stated direction is to move to a client-server model, in which the computation would be done primarily on the server–which Sun expects to sell. While the gross sales of additional units cannot be treated as revenue, the additional profit, H, from such sales can be represented as revenue from the open source activities.

Hardware sales are not the only product that fall into this category. Netscape, for example (now owned by AOL/Time-Warner), has moved to a virally licensed open source model for its browser. Netscape's early business model relied upon the sale of server software and the cheap distribution of browsers to take advantage of the servers–and collecting whatever it could, if anything, for the browsers. Netscape's share of the server market is now negligible, and it has been forced not to charge at all for its browser. Netscape created the Mozilla project (mozilla.org), which creates a base browser under a viral license. As with Sun, Netscape has special rights, and uses a combination of Mozilla and proprietary software to produce its Netscape browser. Additionally, Netscape *removes* some of the features of Mozilla that interest the open source developers but not its business partners, such as the easy blocking of "pop up" advertisements. Even though Netscape does not receive direct revenue V, it is still to Netscape's advantage to have consumers using a Netscape browser, as it opens by default to Netscape's portal page, which is profitable. The additional profits from the portal are H in this case. Additionally, Netscape may collect support fees S for assisting OEM's in including or configuring Netscape on new machines. As AOL uses a derivative of Microsoft Explorer, the continued existence of Netscape provides bargaining power with Microsoft.

Finally, a firm may have internal uses for the software it develops. There may either be no existing software suitable to the firm's purposes, or the firm may find customized software more effective. In such cases, the firm may either pay to have custom software produced or develop new software on its own. In either case, it may choose a closed or open source solution; and the open source solution may be of either a public

or viral nature. In either case, the firm may choose not to release the source to any third parties if it does not release binaries to the outside world. Furthermore, with a public license, it may choose to release binaries and not source. In any event, the firm receives benefit I from the internal use of the software, with costs considered below.

It is possible that a given firm will have more than one of these factors present. The total revenue to the firm from the use of open source software is then

$$R = V + S + H + I \tag{3}$$

4.3. Costs and Strategic Concerns for the Firm of Open Source

Open source is not an exception to the laws of economics, recent dotcom hysteria not withstanding. To continue its existence, a firm must incur expenses, which in the long run must be less than its revenues. Relying upon open source software will incur expenses, possibly even greater than those incurred in a proprietary solution. The goal of the firm is not to maximize revenue, nor to minimize expense, but to maximize the amount by which revenues exceed expenses.

Of particular concern to the firm is the possibility of decreased development costs with open source software, both in development and maintenance: if the source is open and used by others, the firm can take advantage of the development work of others. Weighing against this benefit is the ability of other firms to benefit as well. This introduces strategic considerations from the firm: how will other firms react to the firm's release of the source, and whether the losses L resulting from sales lost to other firms will be greater than the savings in development costs D. The costs to the firm can then be represented as

$$C = D + L \tag{4}$$

The firm will have development costs in any event–the software must be produced. If the firm releases the source, however, outside programmers may take interest either as a hobby, as in the case of Netscape's Mozilla browser, or for business reasons, as with IBM and its support of Apache and Linux. It is possible for outside developers to bear a significant portion of the costs. To date, the tendency has been for firms to assign personnel to existing projects which benefit the firm–or at least have the potential to do so–such as IBM's involvement and Corel's support of the WINE project (which it wished to use to port its existing applications to Linux), and the the corporate project seeking volunteer help (Mozilla) remains the exception. There is no altruism in

Table I. Simple Game for Firm and Public

Firm	Public: Ignore	Public: Contribute
Closed	(0, 0)	(0, 0)
Open	(5, 0)	(7, 5)

the behavior of these firms; the actions are strictly profit-seeking. What is of interest is *why* profit seeking behavior would result in the release of intellectual property to competitors (and the rest of the world)–at first approach, this seems to be counter-productive, particularly in the case of enhancements made by the firm.

While there may be some level of good will involved in the source release, there are fundamental cost savings involved in the release of the code. Of particular note is Apple's Darwin, the operating system that runs underneath its OS X. Darwin is derived from FreeBSD, NetBSD, and the Mach microkernel from Carnegie Mellon University, all of which are under public licenses. It also has technology from NeXT, which was purchased by Apple. It was entirely within Apple's ability to retain the changes made. Nonetheless, Apple released Darwin under a public license. Furthermore, Apple fed large numbers of bug fixes back to NetBSD, in spite of the absence of any obligation to do so. Apple's motivation may be seen in its announcement that it will be "synchronizing" Darwin to FreeBSD (Darwin FAQ): by turning over its own changes, even those that would give it some competitive advantage, Apple's product is "automatically" maintained. If Apple were to make a private change in the code, it would benefit in the short run. In the long run, another change would be made in the the same sections of the "other" source base. As Apple's code would now be different, the change could not be directly made; that is, the opportunity cost of keeping the changes is forgoing the external maintenance. This may increase V and S and decrease L, but at the same time it increases D. Apple has no interest in Darwin in and of itself; it is a necessary component for its OS X product–and a component that Apple tried and failed to develop at least twice.

This simplest case can be represented with very simple game. Table I shows the game for a hardware firm which needs a program as part of its product and faces no direct competitor able to benefit from the code. If the firm chooses a proprietary design, the cost is 10 (payoffs are reported relative to this amount). However, there is a program already available, as in the case of Apple and Darwin, which can be used by the firm, reducing the cost by 5. With no competitor to take advantage of the code, there is no incentive for the firm to not release

Table II. License Choice for the Firm

		Public: Ignore	Public: Contribute
Firm	Closed	(0, 0)	(0, 0)
	Public	(5, 0)	(7, 5)
	Viral	(4, 0)	(6, 3)

the changes (as is the case for a viral license). The only other player is the public, which can either ignore the code, or can contribute and release changes. Between these changes and the fact that the public will provide a portion of the regular maintenance of the software, the firm obtains an additional payoff of 2 when the public chooses to contribute. In this case, the open source option is a dominant strategy–D is lower for the firm regardless of the choice made by the public.

Also of interest in the Apple example are the different choices presented by public and viral licenses. By using a public license, Apple was entitled to keep the code changes, but chose to forgo doing so for strategic reasons. If Apple had used a viral code base, however, not only would it have been forced to release the Darwin code, but would also have had to either take great care to insure that no portion of OS X was a "derived work" of Darwin within the meaning of copyright law (which might or might not have been possible), or release the source to OS X.

License choice may be represented in the more complicated game of Table II. This is an extension of the prior game, with the same costs and benefits. The firm, however, receives an additional choice between a public or a viral license. The payoffs for the public license remain as before. The viral license can only be used by a subset of the public that could use the public license. This is necessarily true, as the use of public code in a viral project is possible with all major licenses in use today, while the converse is not true. Thus the payoff to both the firm and the public is weakly smaller for a viral license (it is possible that the subset of contributors is the full set of potential contributors, in which case equality holds). Table II reflects this by reducing the gains to the firm by 1 for the viral license. Open with a public license remains the dominant strategy.

Finally, the actual choices made by Apple illustrate that open source is not an "all or nothing" proposition. For the commodity portion of the system, which simply needed to be a working Unix-like operating system, Apple chose to release under a public license. For OS X itself, which is part of Apple's competitive advantage, a pure proprietary solution was chosen. Similarly, Darwin runs on x86 hardware, while Apple sees no advantage to releasing OS X for x86–the reduction in

Table III. Firm with License Choice and Potential Competitor (Firm, Public, Competitor)

Firm	Public/Market Competitor: Ignore/ Ignore	Contribute/ Ignore	Ignore/ Hoard	Contribute/ Hoard	Ignore/ Contribute	Contribute/ Contribute
Closed	(0, 0, 0)	***	***	***	***	***
Public	(5, 0, 0)	(7, 5, 0)	(4, 0, 3)	(6, 5, 4)	(5, 0, 3)	(7, 6, 4)
Viral	(5, 0, 0)	(7, 5, 0)	***	***	(5, 0, 3)	(7, 6, 4)

hardware sales would not be fully offset by the additional licensing revenue.

This ties into the possibility that there exists a competitor which *could* gain an advantage from the software developed. Suppose, for example, that Microsoft decided to use a Unix as the basis for its next release of Windows, and that this were to make a Windows machine more desirable as compared to an Apple. The entire benefits supposed so far for Apple relied upon the fact that Unix was a necessary portion of the product, and that Apple's gains came not from the fact that the license was public, but that a public license reduced development costs more than a viral license.

Table III shows a game in which a market competitor can adopt the product (irrelevant choices are blanked from the table with asterisks). With a public license, the competitor may choose to either "hoard" any changes it makes for its own use, or contribute them back to the general project. While the relative sizes of the L, the loss to the original firm from the enhanced competition, and D_m, the savings in development costs if the market competitor contributes its changes, will vary by circumstance, for this example both are assumed to be 1. "Irrelevant" branches of the game are indicated with asterisks for the sake of clarity.[5]

In this game, the firm's best choice is a viral license. If it chooses a public license, the market competitor can take the software, inflicting losses L, with no offset to the firm. The viral license forces the competitor to release changes, which (in this case) yields the same payoff for the firm as if the competitor had ignored the software.

Finally, the more general case of this game is represented in Table IV. Additional notation has been introduced, with D_p and D_m representing the cost savings to the firm from contributions by the public and the market competitor, ΔA representing any increase in administrative costs for monitoring code usage and maintaining "Chinese walls" with a viral license, P being the gain to the public if the project is released

[5] Public payoffs when the firm chooses "Closed" are the same in all cases as for the "Ignore/Ignore" choice. If the market competitor chooses "Hoard" and the firm chooses "Viral", the competitor is forced to make another move–imposing either of the others upon the game would be arbitrary and misleading.

Table IV. General Case for Frim With Competitor(Firm, Public, Competitor)

Firm	Public/Competitor: Ignore/ Ignore	Contribute/ Ignore	Ignore/ Hoard	Contribute/ Hoard	Ignore/ Contribute	Contribute/ Contribute
Closed	$(-D, 0, 0)$	***	***	***	***	***
Public	$(-D, 0, 0)$	$\begin{pmatrix} -D+D_p, \\ P, 0 \end{pmatrix}$	$(-D-L, 0, M)$	$\begin{pmatrix} -D+D_p-L, \\ P, M+D_{mp} \end{pmatrix}$	$\begin{pmatrix} -D+D_m-L, \\ 0, M \end{pmatrix}$	$\begin{pmatrix} -D+D_p+D_m-L, \\ P+P_m, \\ M+D_{mp} \end{pmatrix}$
Viral	$(-D-\Delta A, 0, 0)$	$\begin{pmatrix} -D+D_p-\Delta A, \\ P, 0 \end{pmatrix}$	***	***	$(-D+D_m-\Delta A-L, 0, M)$	$\begin{pmatrix} -D+D_p+D_m-\Delta A-L, \\ P+P_m, \\ M+D_{mp} \end{pmatrix}$

as open source, M being the value to the market competitor of the project if it can hoard its changes, and D_{mp} the gains to the market competitor of contributions from the public.

While several factors are included, it would appear that the most important consideration is the relative size of L and D. Secondarily, there is the special case in which D_m is larger than L–the case in which the value of the competitor's changes to the code base exceed the business lost to the competitor. In this case, the firm actually receives a higher payoff from the viral than the public license.

It should be noted that in cases where L is significant, the product actually yields a competitive advantage. As such, the firm should either choose the proprietary solution, or use special rights license as a "poison pill." This is in fact what appears to happen in the real world: both Netscape's Mozilla and Sun's OpenOffice are licensed such that any competitor would be forced to release its *entire* product as open source, undercutting any serious chance at raising revenue from sales, while Netscape and Sun are able to release special versions (Netscape and StarOffice) with features not available in the open source versions.

5. Open Source as a Quasi-Public Good

Public goods are of interest because while society benefits from the provision of the good, no individual entity receives enough benefit to provide the good. The Nash equilibrium is for all players to be free riders (not provide or contribute to the good), with the result that the good is not made. When software is open sourced, is very close to meeting the definition of a public good: it is is non-excludable, and nearly non-rival. It is not necessarily fully non-rival, as L (defined in Section 4) covers potential losses due to the gains of rivals.

While perhaps not fully non-rival, it is easy to identify cases in which the reduction in D is orders of magnitude larger than L. Darwin appears to be one example, but IBM's support of Apache more clearly meets this qualification. IBM sells web servers. There are not separate markets for web serving hardware and web serving software; rather, there is a market for served web pages. In a competitive or monopolistically competitive market, IBM does not stand to make an additional profit by having separate products–the entire price that IBM can charge for a given system is based upon the capacity and reliability of the entire system. IBM thus has an interest in having a better system available, but is not concerned with *how* the system is improved other than that the cost of doing so be minimized. IBM recognized that rather than maintaining its own software, it could adopt Apache and

Table V. Creation of a Project as a Quasi-Public Good

		Others: Contribute	Others: Private
IBM	Contribute	(3, 5)	(−17, 0)
	Private	(0, −15)	(0, 0)

provide the improvements needed in that product. While Apache has a public license, IBM finds it more profitable to turn the improvements into part of the project than to keep and maintain its own derivative. IBM is not bearing the entire cost of providing the quasi-public good, but the marginal benefit to IBM *of contributing to the good* (releasing the code that it develops) is greater than the cost to IBM of doing so (IBM would incur a large D to maintain the separate product). This is in contrast to the true public good in which the cost of production is small compared to the social benefit, but large compared to the private benefit.

The simple game in Table V shows the incentive to manufacturers to create the quasi-public good. Suppose both agents (IBM and Other) need the program, with a cost of 10 to develop (payoffs are reported relative to this cost). Suppose further that the program is more valuable to IBM than Other. Finally, suppose that the project will be slightly more expensive to develop jointly, reflecting both coordination issues and the fact that the players need different features, bring the total cost to 10. The project still gets built, even if one party must bear a greater While each firm pays less when jointly developing the project, a firm that contributed and was rebuffed would face a greater cost than if it had developed the project itself. Nonetheless, if the contributions proposed are 7 for IBM and 5 for Other, then "Contribute-Contribute" and "Private-Private" are both strong Nash equilibria, and there is no incentive for either to back out of the joint project.

The continued open source development of the project is a likelihood. Table VI demonstrates payoffs for the existing Apache project. "Contribute-Contribute" is a strong Nash equilibrium; each party gains from the contributions made by the other, which can be integrated in a straightforward manner. If one party were to stop contributing, while the other continued to share, *both* parties lose–the locked out party loses all benefits of open source development, while the locking party will have more difficulty integrating the shared changes as its code base will be different.[6] The payoff to the locked out firm is the same

[6] A "patch" is typically a human readable file which gives changes to be made *relative to* the existing code. If the existing code changes, the patch may fail. For

Table VI. The Apache Project as a Quasi-Public Good

		Others	
		Contribute	Close
IBM	Contribute	$(0, 0)$	$(-5, -1)$
	Close	$(-1, -5)$	$(-5, -5)$

whether it shares its code or not, making "Close-Close" a weak Nash equilibrium.

There are also cases in which the initial program is provided entirely or nearly so by a single firm, such as Darwin, but these seem far less common than participation in an existing project. It is too early to tell whether Netscape's Mozilla and Sun's StarOffice will succeed, but in both cases an existing product had its source released under a viral license–a bet by the owner that the private benefit to the owner would exceed the cost of providing the good (including foregone revenue).

6. Conclusions

The use of open source software by the competitive firm, whether as a consumer or producer, is consistent with traditional economic analysis. Firms will consume the software available at the lowest cost, and will participate in the production of commodity components of their product line as a method of reducing costs. Firms may release code that they are entitled to keep private so that it will become part of a maintained code base, which is less expensive than maintaining the code independently.

References

Franke, Nikolaus, and von Hippel, Eric Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software. MIT Sloan School of Management Working Paper #4341-02.

Harhoff, Dietmar, Henkel, Joachim, and von Hippel, Eric, "Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations.v *Working Paper???*

von Hippel, Eric, "Exploring the Open Source Software Phenomenon: Issues for Organization Science," Sloan School of Management, MIT

Johnson, Justin Pappas Economics of Open Source Software. ***

Khalak, Asiv Economic model for impact of open source software. (Dept. of Mech Eng, MIT), akhalak@alum.mit.edu

example, if the patch instructs to put the dog next to the fire hydrant, and the code now has a street light instead of a fire hydrant, the patch will not work.

Lerner, Josh, and Tirole, Jean. Some Simple Economics of Open Source. *Journal of Industrial Economics* 50, 197-234, 2002.

Lecocq, Xavier, and Demil, Benoit, working paper, Open Standard: role of externalities and impact on the industry structure.

Mockus, Audris; Fielding, Roy T.; and Herbsleb, James. A case study of open source software development: the Apache Server. Proceedings of the 22nd internation conference on Software engineering, 263-272, 2000.

Mustonen, Mikko. Copyleft-the economics of Linux and other open source Software. Discussion Paper No. 493, Department of Economics, University of Helinski.

Pal, Nilendu, and Madanmohan, T. R. Competing on Open Source: Strategies and Practice.

Corel's Contributions to the Wine Project. *Linux Today*, June 23, 1999.

Sandberg, Jared. IBM Will Distribute Apache Software for Free, but Charge for Maintenance. *Wall Street Journal*, June 22, 1998, p. B5.

Gomes, Lee. More Big Technology Firms Embrace Broad Use of Linux Operating System. *Wall Street Journal*, May 13, 1999, p. B4.

Buckman, Rebecca. Microsoft Net Rose 12% in Quarter–Profit, Revenue Forecasts Are Reduced Amid Weakness In PC, Software Markets. *Wall Street Journal*, April 19, 2002, p. A3.

Debian Social Contract, http://www.debian.org

http://developer.apple.com/darwin.

http://developer.apple.com/darwin.projects/darwin/faq.html.

General Public License. http://www.fsf.org/gpl.

http://www.mozilla.org.

http://www.netcraft.com/survey, September 2002.

Debian Open Source Definition, http://www.debian.org.

http://www.winehq.org