If Open Source Code Is a Public Good, Why Does Private Provision Work (Or Does It)?

by David R. Agrawal

BROADLY DEFINED, OPEN SOURCE SOFTWARE (OSS) is computer software that has openly available and modifiable code distributed under an open source license. OSS or Free Open Source Software (FOSS)[1] is distributed freely unless the code is bundled with other features, such as installation or service promises. OSS licenses allow consumers to use the software, modify it within certain parameters, and then redistribute the edited software under the same licensed terms.

Last year, the French government debated banning OSS as a matter of innovation and technology policy.[2] Meanwhile, in the United States, the Commonwealth of Massachusetts adopted polices that encourage the utilization of open systems and documents.[3] However, before adopting legislation concerning OSS, policymakers should consider an economic and technical analysis. Is OSS efficiently provided?[4] Moreover, if so, why is OSS provided efficiently through private means?

Knowledge is a public good[5] and the mantra of public economics is that private markets inefficiently provide a public good—in other words, that private markets will be unable to provide the socially optimal level of a public good. Such a rationale implies a role for government intervention because of market failure.[6] For this reason, economists justify a need for the government to subsidize the provision of knowledge by creating intellectual property rights, copyrights, and patent rights, among other options. If such a theory applies to public goods, similar public incentives should be necessary for open source software code.

OSS is a public good and thus must satisfy two criteria—non-

David R. Agrawal is a master of public policy student at the Goldman School of Public Policy, where his research focuses on public finance, public economics, and political economy. He is also a head graduate student instructor in the University of California, Berkeley, Department of Economics. David earned two bachelor degrees from the University of Connecticut, in political science and economics, respectively. *He would like to thank his advisor, Stephen M. Maurer, for his guidance and suggestions.*

rivalry and non-excludability.[7] Software code is an immaterial good: one consumer can use code without interfering with another person's ability to use the same code. Therefore, all software code is, by definition, non-rival. The qualification of non-excludability is slightly more complicated. A software developer can charge a fee to exclude individuals from obtaining all or part of the code to keep trade secrets. Because OSS allows the software code to remain open, all who want to work with the software have free access. Freely available OSS code is non-excludable, and, therefore, not all code meets the second criteria of non-excludability.[8] As a result, only code issued under an open source license agreement is a public good.

No single dominant incentive mechanism can encourage additional innovation of OSS. Rather, programmers supply OSS for reasons that vary depending upon the exact code they are writing. Incentive mechanisms, such as altruism, work only when programmers do not feel exploited for their services. Code is dynamic; when distributed under an open standard, the finished product is ultimately dependent upon the needs of the final consumer. As a result, OSS is a complex, alterable public good, and the motivations of its suppliers are dynamic. No single dominant policy will be effective in encouraging open standards.

For many consumers of OSS, the software is only a semi-finished good that generates little value until the code has undergone revision by the user. Thus, creating the ultimate finished product will require a sequence of motivating incentives.

Based on the fact that OSS is a public good, I will analyze the factors that allow markets to efficiently provide OSS. Initially, I will also address whether or not markets efficiently provide OSS. Further, I will consider whether code programmers[9] are naturally altruistic, motivated by signaling, driven by game theoretic first-mover decisions, or driven by firms' profit incentives. Finally, I will address how these issues influence policymakers who are attempting to create innovation incentives.

LITERATURE REVIEW

I conduct my analysis of the motivations to privately provide OSS within the context of four varying viewpoints. For one, Lerner and Tirole (2002) present the proliferation of OSS as a result of individuals wanting to signal their ability for career advancement. Alternatively, Bitzer, Schrettl, and Schröder (2004) view OSS projects primarily as the product of a need for software, fun for the developer, and a culture of altruism. Another author, Johnson (2002), analyzes the benefits and costs of developing versus not developing and concludes that game theory models predict programmers will develop code with some free riding. Finally, Bessen (2005) views FOSS as a way for firms to maximize profit, when releasing the code as a public good is more profitable than maintaining the software as a private good.

Lerner and Tirole demonstrate that the most sophisticated of software users—those consumers who have needs that go beyond the pre-packaged standard software—are the users of OSS. Accordingly, OSS programmers will develop software if the net benefits are greater than the costs. For example, talented individuals will demonstrate their ability to program by producing OSS. Through this process, OSS serves as a way for individual programmers to highlight or signal a particular talent to employers. In these instances, the advantages of signaling are greater than the opportunity costs of time spent on a project. Lerner and Tirole reject the claim that fun and altruism motivate programmers because such explanations are not applicable to other public goods. Nonetheless, signaling creates a strong incentive for individuals to develop OSS. Proprietary systems will purposefully develop projects with low signaling incentives. Programmers will mostly work on projects with the strongest signaling incentives. This implies that programmers will work on large and visible projects with a high probability of success. Therefore, for signaling to work, early code modifications often require a pre-existing structure to the project in order to demonstrate the project has benefits.[10]

Bitzer, Schrettl, and Schröder counter the arguments of Lerner and Tirole. The authors argue that signaling cannot motivate private provision of OSS because small OSS projects have no signals, yet are still developed. The authors also claim that signaling cannot explain the initial investment of time because each project has a probability of failure.

Thus, Bitzer, Schrettl, and Schröder argue that need for software, fun for the developer, and a gift-giving culture motivate OSS programmers. Accordingly, when facing a strategic decision to develop or not develop, certain types of programmers will have incentive to develop in the current period without delay. The person who decides to develop the software will be young, derives a high benefit from the software, obtains value from gift-giving and fun of code, and faces low costs of development.[11]

Mathematically, Johnson inspects the private value of development to an individual (v_i) and the private cost of development to an individual (c_i). If the ratio v_i/c_i is sufficiently high, the programmer will develop the software. Because the number of programmers influence the probability of development, $v_i - c_i$ must be greater than $\pi^i v_i$, where π^i is the probability that the innovation will occur if individual i does not develop. Yet, as with any public good, free riding dilemmas arise. Although free riding may prevent some projects from being produced, the free riding in this game theoretic model limits the amount of wasteful duplication among products because some programmers will decide not to develop and will wait for someone else to create the program. Accordingly, projects that require all options to be developed are best provided when the number of developers is small. Projects that allow changes and improvements to be made in increments, conversely, are best provided when the number of developers is large. Open source is less likely to work for projects with a large number of tasks to development. Thus, as the number of tasks becomes larger, firms will intervene to develop the project.[12]

Another scholar approaches the question of FOSS by examining the interaction of firms and individuals. Bessen argues that FOSS is not just a public good, but rather a complex public good—complex because FOSS provides a large number of applications for a diverse group of users. Using value-cost analysis, Bessen argues that low-value customers have incentives to self-develop because firms may price pre-packaged software too high. Additionally, individuals with excessively complex needs will also self-develop. Thus, the market for software differentiates between individuals with simple needs and individuals with complex software needs. FOSS can co-exist with profit maximizing firms because the market for software is segmented. Some users will always have needs that are more complex, or they will see the monopoly price of the software as excessively high. For this reason, some firms can maximize profit by developing FOSS in order to meet complex needs. Ultimately, FOSS is a complementary means of development to proprietary production because FOSS is an excessively complex public good.[13]

A New Theory Regarding the Nature of Public Goods for the New Economy

The theories outlined above rely on similar assumptions and models that make the theories compatible, yet contradictory, on a number of points. One critical assumption underlying all four papers is that OSS is efficiently (or relatively efficiently) provided by private markets. However, the key assumptions for three of the papers are that OSS possesses the characteristics of a traditional public good, and that the new economy has not altered the very nature of public goods. I will now address the validity of these two assumptions in order to create an alternative model of OSS. Questioning the second assumption clearly reveals that no dominant incentive exists for OSS.

A new, high-tech economy emerged during the 1990s. Particular goods in the new economy do not obey traditional microeconomic theories. In the technology sector, some goods, such as fax machines, have network effects that alter traditional demand curves.[14] Additionally, some goods in the new economy have initially high fixed costs but have a zero marginal cost for producing the next unit, which especially applies to telecommunications services and information services. As such, some prices cannot equilibrate at marginal costs.[15] The high technology nature of the new economy merits the development of new economic theories for high technology sectors.

The new economy also fundamentally alters the view of new public goods. Unlike traditional public goods, the market for OSS products is much more specific. Under the traditional definition, a person sailing a cargo ship consumes the light from a lighthouse as a public good. However, the sailor does not care about the specific make of the lighthouse—but only that she can see it. The sailor cannot change the lighthouse when she uses it and all sailors will use the same lighthouse to navigate. With OSS, the customer is seeking a specific code that meets her specific use—or, to continue the analogy, the customer cares about the specifics of the code. Now, each software customer will not use the same code to meet his needs. Rather, in a segmented market, the customer selects a specific type of the OSS public good. Furthermore, unlike the

lighthouse, the consumer can also adapt the public good while consuming it—changing the make and color of the code.

If applying traditional theories of public goods, economics will converge on a single innovation mechanism as optimal. Such a convergence of thought would ignore the fact that OSS is not a traditional public good. As such, economists should not use traditional examples of public goods to justify a dominant incentive mechanism for code. The complex nature of OSS must be integrated with the motivations of a diverse group of actors in order to create a new economic theory for a new type of public good.

I set forth a new theory of OSS that does not rely on traditional examples of public goods. Simply comparing OSS to other public goods can be faulty, given the continuously changing nature of code. Bessen begins justifying how OSS is different from most public goods when he writes, "Complexity insures that most of the cost of software arises from testing, debugging, and computer maintenance . . . not from the original design and coding."[16] However, OSS is more than just more complex—code is changeable and individually tailored to a consumer. In order for programmers to privately provide OSS efficiently, it must have certain distinguishing characteristics above and beyond the traditional assumptions of public goods. These distinguishing features include complexity, dynamics, market segmentation, and individual demand-driven creation. Furthermore, OSS is not just one product, but is a sequence of products. For many consumers of OSS, the software is only a semi-finished good that generates little value until the code has undergone revision by the user. Thus, creating the ultimate finished product will require a sequence of motivating incentives.[17]

OSS code is a public good that is non-excludable and non-rival. Nevertheless, OSS has more characteristics than non-rivalry and non-excludability. Code is a changeable good—a public good that the demands of one consumer can change dramatically. Since code is fundamentally dynamic (alterable at low cost), its fixed costs are low. Code is also subject to consumer demands and varying consumer preferences. Although easily changeable, code is complex and requires expertise. The lines of code are parts of a complex whole, but altering one part does not destroy the whole. Finally, the characteristics of a particular software code depend on the interests of a diverse set of consumers, where the suppliers are often also the consumers. Perhaps if economists view OSS in such a context, the reasons underlying private provision will become clearer and more diverse. As economists continue to analyze the private provision of public goods, I believe more emphasis needs to be placed on specifically defining code. Economists cannot view code as a single product with the characteristics of traditional public goods. Nonetheless, if the nature of the public good is, in fact, unique, then the fact that OSS can be provided by private sources may be a result more of its characteristics as a good than the motivation of its creators. OSS is not a traditional public good: it is much more complex, dynamic, market-segmented, and driven by specific consumer demands.

OSS is not a traditional public good: it is much more complex, dynamic, market-segmented, and driven by specific consumer demands.

AN ANALYSIS OF INCENTIVES FOR PROGRAMMERS

The provision of OSS appears efficient (or at least nearly efficient) in private markets.[18] Individually demanded projects are produced if the project has value, and code is not just produced for large powerful demanders. Even small projects with low demand are developed. Computer programmers have not petitioned the government to create additional intellectual property incentives. Each of these signs indicates that the level of open source code is efficient in production. If code is a traditional public good, OSS should be under-provided by private markets. If OSS possesses more distinguishing characteristics than traditional public goods, private markets can provide adequate incentives for the distribution of an optimal level of OSS. Although the quantity of OSS may be efficient, the dominant motivation of programmers is unclear. One single motivating factor cannot induce all individuals to provide OSS. By collectively examining a number of factors such as signaling, altruism, value benefit games, and profit maximization, a theory that no single incentive can efficiently motivate programmers emerges.

The new economy has no dominant form of intellectual property for stimulating a diverse

range of innovations given varying technologies. In other words, no single incentive mechanism will stimulate a wide variety of open source code projects. In the new economy, patents, licenses, and grants are not always the best form of motivation. Individuals are motivated to begin development of different types of products for different reasons that often depend on the characteristic of the good being produced.

Some individuals (perhaps motivated by altruism) will prefer to work on highly visible new projects. Other individuals (perhaps motivated by fun or gift-giving), by contrast, will want to work on small projects or may simply contribute a small amount of time by contributing revisions to an existing project. For example, although signaling incentives may not be important for small projects, other programmers work on larger products because of signaling incentives. Additionally, gift-giving is a form of signaling; thus, the signaling that applies to small OSS projects is simply of a different magnitude. Furthermore, fun is an important incentive because it motivates a wide range of small project programmers, although this enjoyment may not be the primary motive for large projects. Individual firms (perhaps motivated by profit) may seek to provide a complex set of services for a diverse customer population. Some individuals may not be motivated at all because the programmer sees no profit motive or incentive to create OSS. In each case, the project facing the developer is vastly different. Thus, comparing the motivations of a small non-visible project to the motivations driving a major multi-dimensional task such as developing Linux is illogical.[19] Just because certain incentives such as altruism and fun are not viable explanations of traditional public goods, does not mean that they cannot apply to certain OSS in the new economy.

Additionally, the OSS market is segmented into a number of products. Market segmentation allows different solutions to motivate individuals based on the type of project. If OSS is not always the same software, the OSS product of one programmer is a completely different product from that of another programmer. So long as individuals have varying preferences, some individuals may gain greater returns from altruism on small projects as opposed to signaling gains on other time-consuming projects. Market segmentation creates a diverse range of OSS and thus requires a diverse range of programmers to develop the technologies. All four authors' theories can co-exist if the assumption of public good characteristics is fundamentally altered. Private markets can provide OSS because of combinations of signaling, altruism, value benefit games, and profit maximization.

I conclude that open source has no dominant incentive mechanism. Altruism may best motivate small projects. Signaling may best motivate large and visible projects. Profit maximization may best motivate products demanded by large firms. On the other hand, personal recognition or gift-giving incentives may motivate some small projects while not motivating some large ones. If profits, fun, altruism, signaling, or gift giving do not motivate the programmer, simple and economically logical value-to-price comparisons may provide further insight. My argument is based on the claim that varying forms of motivation can co-exist and still provide an efficient (or near-efficient) amount of the public good. Patents, copyrights, prizes, trade secrets, and government contracting can co-exist to provide a near-efficient amount of innovation or knowledge. The same principle holds true for OSS, a new type of public good for the new economy.

In the traditional economy, free riding is something policymakers want to minimize. In this unusual case, free riding may actually improve the efficiency of allocating the public good (by preventing waste) if programmers are sufficiently motivated to provide the good privately in the first place.

CYBERLIFE, THE NEW ECONOMY, AND THE PROBLEMS WITH INCENTIVES

Evidently, a number of reasons exist for programmers to develop OSS. Each of the different sets of motivations comes with economic tradeoffs, including principal-agent problems, free riding, the inability to connect software to demand, the failure to provide information, and weak incentives under certain innovation incentives. For example, signaling clearly works best for some projects and not others, while altruism works for some projects, but

not others. In fact, certain types of intellectual property may fail to create incentives if costs are high or may create excessive incentives under particular circumstances.[20] For example, time constraints, opportunity costs, and skill levels may also restrict individuals to make varying choices.

When value-cost analysis is a successful motive, the game theory outcomes may not be perfectly optimal. Game theoretic models applied to OSS require a sufficiently high level of value, but value is not particularly large for a majority of people. Additionally, the argument requires perfect information about the specific value and costs of a project. In many circumstances, the programmer may misestimate the values or costs as too low or high. Thus, if value-cost analysis is the dominant motivation behind OSS, the government may still have a role to provide additional incentives that enlarge the perceived value or make individuals realize the true value of a project.

While game theoretic models may not adequately confront tradeoffs of perfect information and decisions, they are successful at removing some inefficiency. For example, having a large number of uncoordinated producers with varying incentives could result in wasted duplication inefficiencies. By incorporating free riding into a model of development, high duplication rates are reduced substantially. In the traditional economy, free riding is something policymakers want to minimize. In this unusual case, free riding may actually improve the efficiency of allocating the public good (by preventing waste) if programmers are sufficiently motivated to provide the good privately in the first place. Thus, free riding in the case of OSS results in more, not less, efficient provision of a good. For this reason, OSS once again does not fit the traditional theories of public goods.

Additionally, firms producing OSS as a means of profit maximization alter the value-cost analysis of individuals. Looking at OSS as a complementary good allows for additional private sector provision. In the new economy, a public good, complemented with a private good, can make firms more profitable, although public goods are traditionally not profitable. A firm may release code under an open source license in order to force other companies to a particular product standard or to increase demand for another good. In addition, profit incentives allow for the co-existence of individual and firm development. An individual will develop OSS based on value and price comparisons. Absent profitably manufactured substitutes, programmers decide to produce code after comparing the value and cost. If profitable firms are also producing code, individual programmers analyze the difference between the OSS value and the price of the product as developed by the firm. Programmers compare the value obtained from OSS with how proprietary developed packages are priced. Thus, the individual does not decide to develop OSS based on the cost of developing it, but, rather, based on the opportunity cost of not developing it (the price of proprietary software). The individual may face uncertainty of realizing the true dollar estimate of *c* and may once again inefficiently engage in developing.

If signaling is the dominant motivation, society may be better off by making signaling opportunities more visible in order to encourage more software writing. However, signaling clearly inflicts negative costs on the firms with employees who are engaging in OSS. Firms with employees who work on OSS have incentives to reduce the external visibility of the signals in order to minimize the probability of having valuable employees hired by competing firms. Another limitation of signaling is that individuals are often required to do a certain amount of work before being cited in large projects. Such logic implies that individuals will stop contributing to OSS once a necessary threshold of work is completed. Yet the number of programmers may also influence the ability to complete a project. Once a critical mass of the code is written, even if individuals are shirking after completing a specific amount of work, the project can continue to grow efficiently if the number of programmers is sufficiently large.

[T]he individual does not decide to develop OSS based on the cost of developing it, but, rather, based on the opportunity cost of not developing it (the price of proprietary software).

In cases where altruism and own-need are successful motivating factors, society could have a larger number of small OSS projects if the incentives to gift-giving are increased. The idea of altruism is jeopardized if an individual feels exploited by giving the code freely. Perhaps this idea of exploitation explains why altruism and gift-giving are most easily applied to smaller and less visible projects where exploitation is less likely. In this case, small visible incentives to increase the value of gift-giving may

be essential to encouraging development. Additionally, the own-need motivation requires individuals to place a sufficiently high value on the goods, but does not require other members of society to value the good. This is not to imply that these individual motivations are unimportant, but on their own they may not result in the efficient supply of the public goods demanded.

Markets may not yield the optimal provision of the public good as smoothly as one economic model in isolation may suggest. As such, the actual provision of OSS is not perfectly efficient because in reality, information is imperfect and transaction costs are persistent. The inability to have perfect information about values and costs may result in inefficient provision of the public good. Incentives such as signaling and individual motivation are not fully realized because signaling impacts more people than the programmer. For example, firms seek to reduce the influence of signaling in order to retain employees. If the benefits and costs of signaling were borne entirely by the individual, signaling incentives may be even larger. Other incentive mechanisms come with similar tradeoffs.

Although some inefficiencies exist, programmers have not issued extensive demands for the government to create additional incentives. Furthermore, consumers of OSS are not demanding the government help supply more of the good. Therefore, the provision of OSS must be near (or at) its efficient economic level. The reason OSS is more efficiently provided than expected is that all four economic theories of incentives—signaling, altruism, value-benefit games, and profit maximization—are entirely compatible. Where one incentive mechanism fails or creates inefficiency, another incentive mechanism makes up for the loss. Having explored the role of the market in the provision of OSS, I now turn to the role of the government in OSS development.

HOW CAN (OR SHOULD) LAW GOVERN CODE?

Policymakers and politicians must analyze two questions when developing innovation incentives for code. First, is code efficiently provided by private actors? If not, then the government has some ability to induce more incentive or efficiency. Second, is code a public good in the traditional context? If not, then traditional government solutions to the public goods problem will not work.

Policymakers can consider the overall provision of OSS as efficient. Even if the amount of code developed is inefficient in production and allocation, excessive government intervention likely could stifle creativity and destroy certain incentive mechanisms (such as doing it for fun). Additionally, excessive government intervention may reward large and profitable businesses that produce code, which may decrease the number of small individual actors (and having a large number of innovators is beneficial for incremental projects). Rather than considering the provision of OSS, policymakers should consider the distribution of all code—open and closed source. A correct sequence of interactions between private for-profit actors and free open source providers is necessary to induce efficiency in allocation. In other words, even if the amount of code produced is optimal, policymakers may wish to consider whether resources and production are optimally distributed between profit-seeking producers and open source producers.[21]

Policymakers must also remember that while open source has many merits, open source is not always the optimal solution for all software provision. In reality, some software is best provided under closed-source options. Closed source software is best used to encourage the development of simpler products that can service a wide variety of standardized users. Additionally, closed source appears to be advantageous for large projects that have low startup incentives. Therefore, open source should not be the only intellectual property incentive of the future. Just as OSS has no blanket incentive mechanism, open source should not be the universally acceptable solution for encouraging all types of software design.

Turning to the second question, policymakers must realize that no one dominant policy will be effective in encouraging open standards. Policymakers must adopt policies that are dynamic and provide varying incentives. An omnipotent policymaker would be foolish to establish a blanket policy rule that spurs innovation by one mechanism alone. Rather, a policymaker must realize the complex and varying nature of innovation and fit the intellectual property rule to the appropriate outcome. Any policymaker who seeks to encourage the efficient private provision of OSS as a public good must realize that the incentive system must be just as thorough as the code. Furthermore, policymakers must realize that no universal incentive law can increase the amount of code. The amount of code will increase only if incentives vary depending on the code being developed and the motivations of the programmer developing it. Additionally, policymakers must consider whether more or less free

riding is important for the development of code.

Policymakers cannot apply static incentive mechanisms and expect an optimal level of code. OSS is a privately provided public good because the incentives of writing code are continuously changing according to individual preferences. Policymaking is inherently dynamic and complex—but OSS as a public good is just as dynamic and complex. OSS requires a dynamic, complex, segmented, yet complete solution.

CONCLUSIONS

OSS code is unquestionably a public good. However, this paper argues that the nature of the code itself is a contributing reason to why private markets can provide a public good. No theory of motivation is entirely successful in isolation because the same theories can be applied to traditional public goods without success. If the theories expressed by the four papers I have analyzed do not work for traditional public goods, OSS as a public good has a distinguishing feature that differentiates it from traditional public goods. Bessen indicates that the distinguishing characteristic is that FOSS is a complex public good. In addition to the feature of complexity, I add that code is inherently subject to diverse individual demands of consumers and market segmentation, and that code is inherently dynamic—causing the good to change over time. Individual consumers of code demand different finished products—and often may demand an unfinished open source product so that they can use a part of the product to produce their own good.

This paper attempts to demonstrate that OSS is not really a single public good. Instead, code varies across products in order to meet diverse needs. As a result, the nature of the public good is dynamic and ever-changing subject to a programmer's decisions. Because code is a complex and alterable public good, no single standard rule can optimize the production of code. The motivation of private actors is as diverse as the code itself. As a result, private actors have varying incentives depending upon whether the OSS project is small or large, concealed or visible, simple or complex.

The papers I analyzed operate under two major assumptions. First, open source code is near-efficiently provided. Second, the definition of a public good is not altered in the new economy. In this paper, I question the second assumption by demonstrating that OSS is not a traditional public good—rather, code is dynamic, complex, market-segmented, and subject to individual (not market) demand. Future study may wish to demonstrate quantitatively whether markets efficiently provide OSS. However, the probability of success on open source products seems to indicate that OSS is efficiently provided.

Finally, the paper proposes a course of action for policymakers. Creating laws and policies that are dynamic and that fluctuate across a number of goods is a great challenge. If the necessary outcome is inherently complex, over-simplified policies are likely to do more harm than good. Thus, before creating a universal standard for innovation mechanisms, policymakers must first consider if any merit to intervening in the market exists. If not advantageous, the issue should be left to private markets. However, given near-efficient provision, the government can still enhance the amount of open source code through strategically designed and narrowly applied dynamic solutions. Unlike code programmers, one individual policymaker cannot change laws for a specific use. Thus, the challenge of legislation in the new economy is to design an incentive system that is able to adapt and be applied to the nature of code in a dynamic and complex manner. In a new economy where "code is law,"[22] law must become like code.

LBJ

NOTES

1. I predominantly use the terminology "OSS" throughout the paper except when referring to theories from Bessen (2005), since he uses the terminology "FOSS." Furthermore, according to Bessen, the word "free" implies free modification of code and not a free price. A profit-maximizing firm may then distribute the software at zero price or may distribute the FOSS at some non-zero price. In the second case, the FOSS is no longer a public good. Thus, when referring to firm motives, I only consider situations where FOSS is distributed at zero price (such as when the OSS is an attempt to win a standard war and is not bundled with a private good).
2. FSF France, "French Government Lobbied to Ban Free Software" (November 25, 2005). Online. Available: http://www.fsffrance.org/news/article2005-11-25.en.html. Accessed: December 11, 2005.
3. Information Technology Division of Massachusetts, *Enterprise Open Standards Policy* (January 13, 2004). Online. Available: http://www.mass.gov/Aitd/docs/policies_standards/openstandards.pdf. Accessed: February12, 2006, pp. 1-2; and Information Technology Division of Massachusetts, *Enterprise*

Technical Reference Model – Version 3.5 (September 21, 2005). Online. Available: http://www.mass.gov/Aitd/docs/policies_standards/etrm3dot5/etrmv3dot5informationdomain.pdf. Accessed February 12, 2006, pp. 2-3.

4. The issue of efficient provision of OSS can also be framed with regard to the efficiency of code in general. Desiring the efficient production of OSS is a different problem that achieving the efficient allocation of code. The second problem requires an interaction between private and profitable distribution in addition to free OSS distribution.
5. Arrow, "Economic Welfare," pp. 609-25; Geroski, "Markets for Technology," pp. 90-131; and Mansfield, "Social and Private Rates of Return," pp. 221-240.
6. Rosen, *Public Finance,* pp. 65-69.
7. Ibid., pp. 55-56.
8. Bitzer, "Intrinsic Motivation," p. 2 and Appendix B.
9. Individuals who develop code may not necessarily be programmers, however, this paper will refer to these individuals as "OSS programmers" instead of "OSS writers" as Christian Friesicke references and suggests. Interview with Christian Friesicke in Berkeley, CA, Graduate Student Cand. Ing., Technische Universität Hamburg-Harburg, November 21, 2005.
10. Josh Lerner and Jean Tirole, "Some Simple Economics of Open Source," *Journal of Industrial Economics,* vol. 50, no. 2 (June 2002), pp. 197-234.
11. Bitzer, "Intrinsic Motivation," pp2-22.
12. Justin P. Johnson, "Open Source Software: Private Provision of a Public Good," *Journal of Economics & Management Strategy,* vol. 11, no. 4 (Winter 2002), pp. 637-662.
13. James Bessen, "Open Source Software: Free Provision of Complex Public Goods," Research on Innovation Discussion Paper (July 2005). Online. Available: http://www.researchoninnovation.org/opensrc.pdf#search='Open%20Source%20Software%3A%20Free%20Provision%20of%20Complex%20Public%20Goods'. Accessed: November 2005.
14. Michael L. Katz and Carl Shapiro, "System Competition and Network Effects," *The Journal of Economic Perspectives,* vol. 8, no. 2 (Spring 1994), p. 93; and Scotchmer, Suzanne, *Innovation and Incentives* (Cambridge, MA: MIT Press, 2004), pp. 292-293.
15. Scotchmer, *Innovation,* pp. 294-295.
16. Bessen, "Open Source Software," p. 7.
17. A sequential ordering of incentives further complicates the motivations of individuals economy, however, such a sequence is necessary to consider as Michael Dintenfass and Christian Friesicke of Technische Universität Hamburg-Harburg suggest. Interview with Michael Dintenfass, Associate Professor of History, University of Connecticut, Storrs, CT, March 13, 2006; and Friesicke interview.
18. If the provision of OSS is efficient and if the private market for code clears, then the provision of proprietary code and open source code is productively efficient, but perhaps may not satisfy allocation efficiency.
19. Linux is an open source operating system that has become a significant competitor of Microsoft's operating system, especially on the sever market. Estimates indicate that over 10 million users around the world use Linux and several firms have emerged to sell software for Linux.
20. Nancy Gallini and Suzanne Scotchmer, "Intellectual Property: When Is it the Best Incentive Mechanism?," in *Innovation Policy and the Economy, Vol. 2,* eds. Adam Jaffe, Joshua Lerner and Scott Stern (Cambridge, MA: MIT Press, 2002), pp. 51, 53-56; and Scotchmer, *Innovation,* pp. 58-59.
21. Until this point, this paper has primarily considered efficiency in production (the amount of code produced), but efficiency in allocation (how resources and production are allocated between profit-seeking markets and freely-giving open source markets) is an important policy question as Michael Dintenfass suggests. Dintenfass interview.
22. Lawrence Lessig, *Code and Other Laws of Cyberspace,* (New York, NY: Basic Books, 1999), p. 6.

WORKS CITED

Arrow, Kenneth. "Economic Welfare and the Allocation of Resources for Invention." In *The Rate and Direction of Inventive Activity,* ed. Richard R. Nelson. Princeton, NJ: Princeton University Press, 1962.

Bessen, James. "Open Source Software: Free Provision of Complex Public Goods." Research on Innovation Discussion Paper, July 2005. Online. Available: http://www.researchoninnovation.org/opensrc.pdf. Accessed: November 2005.

Bitzer, J., W. Schrettl, and P. J. H. Schröder. Intrinsic Motivation in Open Source Software Development. Free University of Berlin Discussion Paper No. 2004/19, September 2004. Online. Available: http://econwpa.wustl.edu/eps/dev/papers/0505/0505007.pdf. Accessed: November 2005.

Dintenfass, Michael. Associate Professor of History, University of Connecticut, Storrs, CT. Interview, March 13, 2006.

Friesicke, Christian. Graduate Student Cand. Ing., Technische Universität Hamburg-Harburg. Interview in Berkeley, CA, November 21, 2005.

FSF France. "French Government Lobbied to Ban Free Software." (November 25, 2005). Online. Available:

http://www.fsffrance.org/news/article2005-11-25.en.html. Accessed: December 11, 2005.

Gallini, Nancy, and Suzanne Scotchmer. "Intellectual Property: When Is it the Best Incentive Mechanism?" In *Innovation Policy and the Economy, Vol. 2*, eds. Adam Jaffe, Joshua Lerner and Scott Stern. Cambridge, MA: MIT Press, 2002.

Geroski, Paul. "Markets for Technology: Knowledge, Innovation, and Appropriability." In *Handbook of the Economics of Innovation and Technological Change,* ed. Paul Stoneman. Oxford: Blackwell Publishers, 1995.

Katz, Michael L. and Carl Shapiro. "System Competition and Network Effects." *The Journal of Economic Perspectives,* vol. 8, no. 2 (Spring 1994), pp. 93-115.

Information Technology Division of Massachusetts. *Enterprise Open Standards Policy* (January 13, 2004). Online. Available: http://www.mass.gov/Aitd/docs/policies_standards/openstandards.pdf. Accessed: February12, 2006.

Information Technology Division of Massachusetts. *Enterprise Technical Reference Model – Version 3.5* (September 21, 2005). Online. Available: http://www.mass.gov/Aitd/docs/policies_standards/etrm3dot5/etrmv3dot5informationdomain.pdf. Accessed February 12, 2006.

Johnson, Justin P. "Open Source Software: Private Provision of a Public Good." *Journal of Economics & Management Strategy,* vol. 11, no. 4 (Winter 2002), pp. 637-662.

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

Lessig, Lawrence. *Code and Other Laws of Cyberspace.* New York, NY: Basic Books, 1999.

Mansfield, Edwin, John Rapoport, Anthony Romeo, Samuel Wagner, and George Beardsley. "Social and Private Rates of Return from Industrial Innovations." *The Quarterly Journal of Economics,* vol. 91, no. 2 (1977), pp. 221-240.

Rosen, Harvey S. *Public Finance.* Boston, MA: Irwin/McGraw-Hill, 1999.

Scotchmer, Suzanne. *Innovation and Incentives*. Cambridge, MA: MIT Press, 2004.