PUBLIC SUBSIDIES FOR OPEN SOURCE? SOME ECONOMIC POLICY ISSUES OF THE SOFTWARE MARKET

Klaus M. Schmidt[†] and Monika Schnitzer[‡]

TABLE OF CONTENTS

I. INTRODUCTION

The open-source movement has been very successful in developing software products such as Linux, Apache, and Sendmail that are now serious competitors to well-established proprietary software. It is attracting a lot of attention not just in the computing community, but also in the media, by academic economists, and most recently, by politicians. In many different countries, there are political initiatives trying to get public support for open source such as direct subsidies to open-source projects, standards for use of open-source software in government agencies, and replacement of some proprietary software with open-source software in schools and universities.[1]

The body of academic literature on open source is small but rapidly growing. Most of this literature tries to understand the governing structure of open-source projects.[2] Why do programmers voluntarily contribute to the public good of open source, even if there are no direct financial rewards? Why have some projects attracted more voluntary support by programmers than others? Why do large corporations such as IBM, HP, and Intel contribute significant capital investments to open-source projects? What business models are likely to be successful in an open-source market?

The question of the economic merits of direct and indirect public subsidies for open-source projects has received very little attention in the literature.[3] This paper tries to fill this gap. We use modern economic theory to analyze specific aspects of the software market in general and of the market for open-source software in particular. Drawing on this analysis, we develop a framework in which the implications of governmental support for open-source projects can be discussed.

In Section II, we briefly describe what is meant by "open source," what objectives and motivations drive the developers of open-source software, and how this software is protected by different types of li-

---

1. See §IV for descriptions of several such initiatives.

2. *See generally* James Bessen, *Open Source Software: Free Provision of Complex Public Goods*, Research on Innovation, *at* http://www.researchoninnovation.org/opensrc.pdf (last visited Feb. 7, 2003); David S. Evans, *Is Free Software the Wave of the Future?*, THE MILKEN INST. REV., Fourth Quarter 2002, at 32; Dietmar Harhoff et al., *Profiting from Voluntary Information Spillovers: How Users Benefit by Freely Revealing Their Innovations* (July 2000, revised May 2002) (Sloan Working Paper), *at* http://web.mit.edu/evhippel/www/FreeRevealWP.pdf; Justin Pappas Johnson, *Economics of Open Source Software*, *at* http://opensource.mit.edu/papers/johnsonopensource.pdf (May 17, 2001); Josh Lerner & Jean Tirole, *Some Simple Economics of Open Source*, 50 J. INDUS. ECON. 197 (2002). Many additional working papers can be downloaded from http://opensource.mit.edu.

3. A notable exception is David S. Evans & Bernard Reddy, *Government Preferences for Promoting Open-Source Software: A Solution in Search of a Problem*, *at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=313202 (May 21, 2002), which examines instances of government intervention in support of open-source software.

censes. In Section III, we discuss potential failures of the software market. This market differs from standard textbook markets in three important respects: (i) it is characterized by large economies of scale, (ii) innovations and rapid technological progress are crucially important, and (iii) there may be strong direct and indirect network effects and high switching costs for consumers. Each of these characteristics may give rise to inefficient market outcomes. We analyze the differences between proprietary software and open-source software with respect to these potential market failures and ask whether open source can mitigate them. In Section IV, we discuss the economic merits of various policies of direct and indirect governmental support for open source, some of which are already being implemented by several countries. Section V concludes.

II. WHAT IS OPEN SOURCE?

Open-source software ("OSS") is software for which the source code is open, that is freely available, to the public: everybody has the right not only to use the software, but also to extend it, to adapt it to his or her own needs, and to redistribute the original or modified software to others. Source code is written in a computer language such as Java, C, or C++, which is easy to read for an experienced programmer. However, before it can be processed by a computer, it has to be compiled, i.e., translated into machine code, which is just a sequence of zeros and ones. This machine code is very difficult to read for humans, and also difficult and time-consuming to retranslate into source code. Therefore, open source requires the free distribution of not just the machine code but also the source code. Given the ready availability of the source code for OSS, firms generally can charge only low prices for such software. Because any recipient of the source code can freely redistribute the software, prices are driven down toward average distribution costs for OSS.

In contrast, a license for "proprietary software" is sold like any other good or service. Because a firm that develops such software wants to make a profit, it must protect its intellectual property rights. The creator of a software program can obtain a copyright and (in some countries) a patent that enables her to prevent others from copying or modifying her work. However, copyrights and patents are imperfect because it is often possible to circumvent them by modifying the software without violating the legal rights of the owner. Thus, for the protection of intellectual property rights in the software industry, it is at least equally important to maintain the "trade secret" of how the software works. Therefore, most commercial software packages contain only the machine code while the source code is kept secret. As Lerner and Tirole note, "When the source code is made available to

other firms . . . , it is typically licensed under very restrictive conditions."[4]

OSS must also be distinguished from "freeware" and "shareware." Freeware is distributed for free, but users do not get access to the source code and are not allowed to modify or extend the software. The same holds for "shareware," which is often offered for free for a trial period or in a "light" version, so that consumers can try out the software before they buy it. There is a great deal of freeware and shareware available, including such well-known products as the Adobe Acrobat Reader.

A variety of motivations and objectives underlie the open-source movement. For example, the Free Software Foundation ("FSF"), whose founder and most prominent speaker is MIT's Richard Stallman, argues that "free software" is, like "free speech," a moral principle: "Free software is a matter of freedom: people should be free to use software in all the ways that are socially useful."[5] Restricting the use of software and not sharing the source code is unethical, and the ultimate goal is "that all published software should be free software."[6] Other participants in the open-source movement have a more pragmatic point of view: "When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves" and "this rapid evolutionary process produces better software than the traditional closed model."[7] They want to make this case to the commercial world and are prepared to cooperate with proprietary software developers in order to foster software development.

Despite the freely distributed nature of OSS, intellectual property protections are critical to the open-source movement. If OSS lacked such protections, anybody could take the software, modify it, obtain a copyright for the modified version, and exclude others from using it. Many licenses have been developed that restrict such commercialization of open-source code. The Free Software Foundation authored one of the first of them, the "GNU General Public License" ("GPL"). Under the GPL, software distributors must make the source code available to recipients of the software and may not restrict the rights of recipients to make the software and source code available to others.[8] Furthermore, all enhancements of the code, including code that combines the OSS with some other, separately developed software, must

---

4. Lerner & Tirole, *supra* note 2, at 200 n.5.

5. *Philosophy of the GNU Project*, Free Software Foundation, *at* http://www.fsf.org/philosophy/philosophy.html (last visited Feb. 9, 2003).

6. Richard M. Stallman, Speech at New York University (May 29, 2001), *at* http://www.gnu.org/events/rms-nyu-2001-transcript.txt.

7. Open Source Initiative, *at* http://www.opensource.org/ (last visited Feb. 9, 2003).

8. *See GNU General Public License* §§ 3, 6, Free Software Foundation, *at* http://www.fsf.org/copyleft/gpl.html (June 1991). The FSF calls this feature of the GPL (and similar licenses) "copyleft." *See What Is Copyleft?*, Free Software Foundation, *at* http://www.fsf.org/copyleft/copyleft.html (last visited Feb. 9, 2003).

be licensed according to the terms of the GPL.[9] Thus, the GPL is "viral" in the sense that any program making use of code covered by the GPL becomes covered by the GPL as well.

The Berkeley Software Distribution ("BSD") license exemplifies a more liberal approach.[10] A user who redistributes a BSD-licensed program, whether modified or not, must retain the copyright notices in the program code, but faces no other serious restrictions. Software licensed under BSD is not viral and does not turn proprietary software that builds on or interacts with it into open source. The Open Source Initiative ("OSI") publishes a list of other frequently used licenses and indicates the types of terms a software license should (and should not) contain in order to be considered open source.[11]

III. POTENTIAL FAILURES IN THE SOFTWARE MARKET

A. Economies of Scale: Static Efficiency

In a static market (with a given technology and no innovations) the standard welfare measure is the sum of consumer surplus and producer surplus.[12] Total surplus is maximized if goods are priced at marginal cost. If the minimum efficient scale is small compared to the size of the market, the competitive price in a long-run equilibrium with free entry equals marginal cost, which in turn equals average cost. Under such conditions, all firms will make zero profits (which include a risk-adjusted normal return to capital). However, the production of any substantial piece of software requires a significant fixed cost (sometimes called "first copy cost") in order to develop the software, whereas the costs of duplication and distribution are typically very small and constant or decreasing with the level of output. For large volumes, the largest component of marginal cost may be support costs, which may have both fixed and constant marginal components. Thus, the production of a given software product is charac-

---

9. The GPL requires: "You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License." *GNU General Public License* § 2, Free Software Foundation, *at* http://www.fsf.org/copyleft/gpl.html (June 1991). See Evans & Reddy, *supra* note 3, at 25–26, for a more detailed critical discussion of the GPL.

10. *See BSD Today: Your Daily Source for BSD News and Information*, *at* http://www.bsdtoday.com; *Open Source Initiative*, *at* www.opensource.org.

11. *The Approved Licenses*, Open Source Initiative, *at* http://www.opensource.org/licenses/index.html (last visited Feb. 9, 2003); *The Open Source Definition*, Open Source Initiative, *at* http://www.opensource.org/docs/definition.html (last visited Feb. 13, 2003).

12. Strictly speaking, this measure is appropriate only if there are no income effects. However, even if there are non-zero income effects, the measure of consumer surplus obtained from a Marshallian demand function serves to approximate the actual welfare change closely. *See* Robert D. Willig, *Consumer's Surplus Without Apology*, 66 AM. ECON. REV. 589 (1976).

terized by economies of scale, and the market for a given software product is a "natural monopoly."[13] Therefore, static efficiency requires that the whole market be served by just one firm, and that this firm charge prices equal to marginal costs.

From a static point of view, OSS has the advantage that it is distributed for free (or for a small fee reflecting distribution costs), which is socially efficient. A proprietary software developer cannot price at marginal cost. If he does not charge at least average cost, where average cost includes a risk-adjusted normal return on capital, he will make expected losses. Of course, a profit-maximizing software developer would like to charge higher prices. However, even if she serves the market as a monopolist, it is not necessarily the case that prices significantly above average cost can be sustained. There are several reasons why a monopolist may not be able to achieve supra-normal profits:

- Even if competition within the market is not possible, there may be competition for the market.[14] If there are no barriers to entry, competition for the market forces the incumbent monopolist to price at average cost in order to prevent the market from being taken over by a competitor. So, the crucial question is how large barriers to entry are in the software market. The fixed costs that are necessary to develop a software product are largely sunk. Thus, the incumbent monopolist may price below average cost if entry occurs, which may eliminate profits for entrants and deter potential entrants. Also, there may be network effects (especially if the products offered by the incumbent and by the entrant are not compatible) and switching costs for consumers, which may prevent them from adopting the entrant's product even if it is priced below the incumbent's software.[15] On the other hand, rapid technological progress in the software market may facilitate entry. The software industry has seen many instances where new products offered by new companies have driven incumbent products with large market shares out of the market. Therefore, the threat of entry does impose some restrictions on the pricing policy of incumbent software producers.
- Software is a durable good. Once installed, it could in principle run forever, and it "wears out" only due to technological

---

13. Note that increasing returns to scale imply here that the cost function is "strictly subadditive," which is the formal requirement for a "natural monopoly." *See* William J. Baumol et al., Contestable Markets and the Theory of Industry Structure 17 (1982).

14. *See* Harold Demsetz, *Why Regulate Utilities?*, 11 J.L. & Econ. 55, 56–58 (1968); Baumol et al., *supra* note 13.

15. *See* discussion *infra* §III.C.

change. Thus, once the monopolist has sold the software at a high price to those consumers who value it most, she has an incentive to lower the price in order to sell to others who are willing to pay more than her marginal cost. So, over time, prices will tend to fall until they reach marginal cost. If consumers anticipate this effect, they will wait in order to buy at lower future prices. The Coase Conjecture suggests that such behavior may force the monopolist to lower her price very rapidly and to sell at close to marginal cost almost instantaneously.[16] However, the extreme result suggested by the Coase Conjecture is unlikely to apply to the software market. For example, if there is a constant flow of new customers with high valuations entering the market, the monopolist's incentive to lower her price is reduced.[17] Furthermore, if there is rapid technological progress and the monopolist continues to improve her product, then the software becomes less durable and "wears out" more quickly. Therefore, old customers with high valuations may be willing to upgrade at high prices. Again, this reduces the monopolist's incentive to lower the price. Finally, sometimes the monopolist can use contractual provisions in order to commit not to lower her price in the future.[18]

- The monopolist often faces competition from a large stock of older versions of her own software. Consumers will not buy a new version if its cost exceeds the value added to the older version.

To summarize, there are several constraints on the pricing policy of a proprietary software developer that may induce her to charge prices significantly lower than monopoly prices. Nevertheless, OSS is more efficient from a static point of view because it is priced at marginal cost (i.e., sold at distribution cost or given away for free).

B. Incentives to Innovate: Dynamic Efficiency

In a dynamic market (one characterized by rapid technological change), efficiency requires not only that goods are sold to all consumers willing to pay more than marginal cost, but also that firms

---

16. *See* Ronald H. Coase, *Durability and Monopoly*, 15 J.L. & ECON. 143, 143 (1972); Faruk Gul et al. *Foundations of Dynamic Monopoly and the Coase Conjecture*, 39 J. ECON. THEORY 155, 168–69 (1986).

17. *See* Joel Sobel, *Durable Goods Monopoly with Entry of New Consumers*, 59 ECONOMETRICA 1455, 1463 (1991).

18. For example, she can rent rather than sell the software to her customers. In this case, if she lowers the price, she has to do so for all customers, not just for new ones. Much software can in effect be "rented" through the payment of periodic license or maintenance fees.

have the right incentives to innovate efficiently. In such a Schumpeterian world,[19] innovation is a high stakes activity because only those firms who are more successful innovators than their competitors will survive. Firms engage in costly innovative activities only if they can protect their intellectual property and only if they can compensate for the costs of failed innovations by charging more than average cost for successful ones.

Clearly, the market for software has changed very rapidly over the last twenty years. Spurred by enormous technological progress in the production of computer hardware, many new and more sophisticated software products have been developed. Many software products that dominated the market in the 1980's and early 1990's have been leapfrogged by superior offerings from competitors, and are now almost forgotten. Some companies maintained or extended their market position by continuously improving the design and functionality of their software products, and by adding many new features. Thus, there can be no doubt that innovation and technological progress play a crucial role in the development of the software market.

A profit-maximizing firm would have an incentive to innovate efficiently if the expected profit from an innovation equaled its expected net social gain. In reality, this is very unlikely to be the case. The literature on the incentives for research and development ("R&D") and innovation has pointed out three external effects that may distort innovation incentives:[20]

- The consumer surplus effect stems from the fact that the innovating firm cannot perfectly price discriminate and therefore cannot capture the entire increase of consumer surplus generated by an innovation. This is a positive externality of R&D that tends to reduce the incentive to innovate.
- The R&D effect arises if a new technology developed in one market may be usefully applied to other markets. It is a positive externality stemming from positive technological spillovers on other markets and technologies. To the degree that the innovator is unable to fully control the use of her new ideas by other firms, this effect also tends to reduce innovation incentives.
- Finally, the business-stealing effect occurs when introduction of a superior technology makes some existing products less

---

19. JOSEPH SCHUMPETER, CAPITALISM, SOCIALISM AND DEMOCRACY 110 (1st Harper Colophon ed., Harper & Row 1975) (1942).

20. *See e.g.*, Paul M. Romer, *Endogenous Technological Change*, 98 J. POL. ECON. 71 (1990); GENE M. GROSSMAN & ELHANAN HELPMAN, INNOVATION AND GROWTH IN THE GLOBAL ECONOMY 105–06 (1991); Phillipe Aghion & Peter Howitt, *A Model of Growth Through Creative Destruction*, 60 ECONOMETRICA 323 (1992).

attractive and therefore takes away some of the revenues that were previously earned by producers of those products. This externality is negative. Note that there is a private but no social return from this revenue-shifting, so the business-stealing effect tends to induce too much R&D.

The net effect of these three externalities is ambiguous. It is possible to construct examples where the business-stealing effect outweighs both the consumer surplus and the R&D effects, so that incentives to innovate are too strong.[21] However, it is generally believed that the most typical situation is for the overall externality to be positive, so that incentives to innovate are too weak.[22] This is particularly plausible in the software market, where the protection of intellectual property rights is difficult.

What are the incentives of programmers to contribute to the development of OSS? At first glance, it seems puzzling that OSS exists at all. After all, there are no direct pecuniary incentives to develop software that will be freely distributed. Some commentators have argued that the open-source community is a "gift culture" that is motivated by altruism and reciprocity.[23] According to this view, people contribute to the public good of open source because they enjoy being part of the open-source community, helping others, and reciprocating to those who have helped them. However, it is not clear why being "part of the community" is more exciting with open source than with any other industry.[24] Furthermore, while it is now widely acknowledged that reciprocity plays an important role in small group interactions,[25] it is very unlikely that altruism and/or reciprocity provide sufficient incentives to explain activity on such an enormous scale; we do not observe similar patterns of behavior in most other areas of economic activity.[26] So what additional incentives motivate programmers to write OSS?

Many contributors to OSS are sophisticated users. For them, the cost of fixing a bug, customizing a piece of software to their own needs, or developing a modest new application is often fairly small. If they have access to the source code, it is relatively easy to spot weak

---

21. *See* Aghion & Howitt, *supra* note 20.

22. For example, in the model by Romer, *supra* note 20, the consumer surplus effect and the business-stealing effect just cancel out, so only the positive R&D effect remains.

23. *See* Eric S. Raymond, *Homesteading the Noosphere*, 3 FIRST MONDAY 10.2 (Oct. 5, 1998), *at* http://www.firstmonday.dk/issues/issue3_10/raymond/.

24. *See* Lerner & Tirole, *supra* note 2, at 217 n.22.

25. See Ernst Fehr & Klaus M. Schmidt, *Theories of Fairness and Reciprocity – Evidence and Economic Applications*, *in* ADVANCES IN ECONOMIC THEORY, EIGHTH WORLD CONGRESS (Matthias Dewatripont et al. eds., forthcoming 2003), *at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=255223, for a survey on the recent theoretical and experimental literature on this topic.

26. *See id.*

points and to write new code for improvements. The opportunity cost of sharing the new code with others is also small. If the improvement is modest, it is not worthwhile to copyright or otherwise protect the idea and try to sell it. Furthermore, the Internet provides a very efficient and inexpensive way to make the innovation accessible to the public. So why not share it with other colleagues?[27]

While this argument can explain why bug-fixing and small improvements are provided by thousands of programmers from all over the world, it cannot explain why some people devote an enormous amount of time and effort in developing a major improvement or a substantial new program. Like the writing of commercial code, this has to be done by a small group of people who coordinate their efforts and work closely together.[28] Lerner & Tirole argue that there are strong signaling incentives that can explain this behavior.[29] A programmer who solved a difficult problem or contributed an important new piece of software signals her outstanding abilities to the outside world. She is recognized by her peers, may get better future job offers, may be invited to participate in commercial open-source projects, or may have better access to the venture capital market if she wants to start her own business.[30] Economic theory suggests that this signaling incentive is stronger the more visible the performance of the programmer and the more informative the performance is about her talent.[31]

---

27. *See* Johnson, *supra* note 2, for a formal game-theoretic model of the incentives to voluntarily contribute to the public good of open-source software. The open-source movement is not the only user-innovation community. See ERIC VON HIPPEL, THE SOURCES OF INNOVATION 11–26, 28–30, 93–100, 108–15 (1988), and Harhoff et al., *supra* note 2, § 2.2.1–2.2.3, for many other examples of industries in which customers frequently contribute to the development of new products and production techniques on a voluntary basis. However, in all of these industries, user-innovation communities and proprietary innovation coexist. Harhoff et al. point out several additional incentives to freely reveal innovations. In particular, a user who made an innovation may reveal it to the manufacturer in order to induce her to adopt the innovation and improve on it. *See id.* § 2.2.2–2.2.4. Furthermore, by freely revealing an innovation, a user may induce other users to adopt the solution as well, which may set a new "standard" that in turn induces other manufacturers to provide complementary product improvements. *See id.* § 2.2.1. These incentives are more likely to work if there is no competition between users. *See id.* § 2.1.

28. One study reports that 83% to 91% of changes in the code of the Apache server software have been written by just 15 top developers. *See* Audris Mockus et al., *A Case Study of Open Source Software Development: The Apache Server*, *in* PROCEEDINGS OF THE 22ND INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (2000), http://portal.acm.org/citation.cfm?doid=337180.337209.

29. Lerner & Tirole, *supra* note 2, at 213–14.

30. These delayed pecuniary incentives are partially acknowledged by open-source advocates. For example, Eric Raymond writes: "It can get you a better job offer, or a consulting contract, or a book deal." Raymond, *supra* note 23. But, according to Raymond, such benefits are "at best rare and marginal for most hackers." *Id.*

31. *See* Lerner & Tirole, *supra* note 2, at 214; Bengt Holmström, *Managerial Incentive Problems: A Dynamic Perspective*, 66 REV. ECON. STUD. 169 (1999).

The signaling account is consistent with several empirical observations. First, giving credit to programmers is strongly emphasized in the open-source movement. Most projects recognize all contributors on their website and highlight the contributions of the most committed programmers.[32] Raymond points out that "[r]emoving a person's name from a project history, credits or maintainer list is absolutely *not done* without the person's explicit consent," and that "[s]urreptitiously filing someone's name off a project is, in cultural context, one of the ultimate crimes."[33] Second, most open-source programs are written for sophisticated users who can evaluate the difficulty of the task and the importance of the contribution. In particular, server operating systems and server applications have done very well, while far fewer consumer applications are written as open source. Third, most open-source projects have a modular structure. This not only facilitates the cooperation of many independent programmers who may be spread all over the world, but also highlights the individual contribution of each programmer. Finally, there is substantial evidence that contributing to open source has helped many programmers gain access to venture capital and receive attractive job offers from commercial software developers.[34]

Finally, some programmers devote time to the development of OSS as part of their employment with commercial software and hardware companies. These companies have an incentive to invest in the development of OSS that complements their commercial products. Some firms, such as Red Hat, VA Linux, and SuSE, provide services and products that are complementary to Linux but that are not efficiently supplied by the open-source movement.[35] If Linux is improved and becomes more widely adopted, these firms gain access to a larger market[36] Similarly, since server hardware and server operating systems are complements, server vendors will realize increased profits if an open-source operating system like Linux takes market share from the proprietary standard, Microsoft Windows.[37] Because companies cannot prevent others from benefiting from its investment in open-source, companies have a strong incentive to free ride on the contribu-

---

32. *See* Lerner & Tirole, *supra* note 2, at 218.

33. Raymond, *supra* note 23. *See also The Open Source Definition* § 4, Open Source Initiative, *at* http://www.opensource.org/docs/definition.html (last visited Feb. 13, 2003) ("[A]uthors . . . have [the] right to . . . protect their reputations.").

34. For example, Lerner & Tirole report that "the founders of Sun, Netscape, and Red Hat had signaled their talent in the open source world." Lerner & Tirole, *supra* note 2, at 218. See *id.* at 219, for a tabular summary of the commercial roles of several other open-source participants.

35. *See id.* at 225.

36. *See id.*

37. *See id.* Firms employ several other more or less successful open source strategies. *See id.* at 225–27.

tions of others to open source, and their subsidies to OSS development are likely to remain limited.[38]

Comparing the incentives to innovate for proprietary and OSS developers, the open-source movement exploits some incentives in ways proprietary software developers cannot readily mimic. For example, because commercial developers do not make source code publicly available, it is impossible for users to fix bugs or customize the software to their own needs. The open-source mode, in contrast, allows users to act on their incentives to improve the software. So, by making the source code available, open-source facilitates low-cost, large-scale, parallel innovation.[39] Furthermore, as Lerner & Tirole point out, signaling incentives are greatly diminished in the proprietary development context:

> [C]ommercial companies will never be able to fully duplicate the visibility of performance reached in the open source world. . . . [They] do not like their key employees to become highly visible, lest they be hired away by competitors. But, to a large extent, firms also realize that this very visibility enables them to attract talented individuals and provides a powerful incentive to existing employees.[40]

However, comparing the private benefits accruing to OSS developers to the increase in social benefits (i.e., consumer surplus) that can be generated by innovative new software, it is apparent that the incentives for innovation are far too small in the open-source mode.

A major advantage of the proprietary mode is that it allows software developers to capture at least some of the fruits of their efforts, i.e., to turn at least some of the consumer surplus generated by their innovations into profit. As in all other industries, the profit motive provides a very powerful incentive to innovate that is not present in the open-source world. As Schumpeter pointed out:

> It is . . . quite wrong . . . to say, as so many economists do, that capitalist enterprise was one, and technological progress a second, distinct factor in the observed development of output; they were essen-

---

38. *See id.*

39. Some commercial software companies try to partially mimic open source by promoting "widespread code sharing within the company," *Id.* at 224, and by making the source code available to some core customers (under restrictive conditions).

40. *Id.* at 223.

tially one and the same thing or, as we may also put
it, the former was the propelling force of the latter.[41]

Twenty years later, Schmookler summarized his empirical findings on
what drives technological progress:

> Despite the popularity of the idea that scientific dis-
> coveries and major inventions typically provide the
> stimulus for inventions, the historical record of im-
> portant inventions in petroleum refining, paper mak-
> ing, railroading, and farming revealed not a single,
> unambiguous instance in which either discoveries or
> inventions played the role hypothesized. Instead, in
> hundreds of cases, the stimulus was the recognition
> of a costly problem to be solved or a potentially prof-
> itable opportunity to be seized; in short, a technical
> problem or opportunity evaluated in economic
> terms.[42]

There is no reason to believe that the software industry differs
fundamentally from all other industries when it comes to incentives
for innovation. If it had not been possible to appropriate the profits of
risky software development projects, the rate of technological pro-
gress in the software industry over the past decades would have been
much lower.

Incentives to innovate must be assessed with respect to not only
the amount of effort and investment they promote, but also the direc-
tion of innovation and its responsiveness to consumer demand. A pro-
prietary software developer has a strong incentive to respond to the
needs of all potential users of his software. The larger the group of
consumers who value his software, and the greater the magnitude of
their valuations, the more copies he can sell and the more he can
charge. A proprietary software developer thus has a strong profit mo-
tive to identify the needs of consumers through market research and to
develop software that satisfies those needs.

While the open-source mode allows sophisticated users to de-
velop and customize software to suit their own purposes, it provides
no incentive to cater to mass consumer markets. OSS developers are
sophisticated users and IT professionals who respond to their own
needs for functional software and enjoy challenging programming
tasks. They also respond to the needs of, and seek recognition from,
other sophisticated users and IT professionals. Writing software for
unsophisticated end users involves many tasks that may be intellectu-

---

41. Schumpeter, *supra* note 19.

42. JACOB SCHMOOKLER, INVENTION AND ECONOMIC GROWTH 199 (1966).

ally unsatisfying (such as providing a user-friendly interface or de-
tailed and easy-to-read documentation) and that are poorly suited to
gaining peer recognition.[43]

Furthermore, developing software for the consumer market re-
quires detailed knowledge of the preferences and practices of end us-
ers. For example, in order to write good accounting software, it is
necessary not only to have good programmers, but also in-depth
knowledge of the rules and requirements of accounting. Such pro-
gramming requires the close collaboration of IT professionals with
accountants and experts in several other fields. Similarly, the devel-
opment of a sophisticated computer game requires the collaboration of
programmers with graphic designers, marketing experts, etc. Even the
development of office software, such as a word processor or spread-
sheet application, requires detailed research on the needs and prefer-
ences of the least sophisticated users. In contrast to proprietary
development, the open-source mode provides insufficient incentives
to engage in such activities.

Thus, it is not surprising that the open-source movement has been
most successful in the development of operating systems and server
application software that respond directly to the needs of IT profes-
sionals and other computer experts, while it has been much less suc-
cessful in developing end user applications. To be sure, there are some
end user applications that are licensed as open source. However, the
most successful of these applications have been developed in the pro-
prietary mode and released as open source only after failing commer-
cially (e.g., the Star Office Suite and the Netscape Internet browser).

C. Network Effects and Switching Costs

It is often argued that strong network effects and switching costs
cause market failures on the software market, and that these character-
istics restrict competition.[44] In this Section, we give a brief summary
of the main effects that can arise if network effects and/or switching
costs are present. As we will see, the problems that arise apply largely
to both proprietary and OSS.

1. Network effects

Network effects arise if there is a complementary relationship be-
tween the adoption of a good by different customers: additional adop-
tion makes existing users better off (i.e., increases their *total utility*

---

43. *See* Lerner & Tirole, *supra* note 2, at 217 & n.23.
44. See Joseph Farrell & Paul Klemperer, *Coordination and Lock-in: Competition with
Switching Costs and Network Effects* 44–45, *at*
http://www.nuff.ox.ac.uk/users/klemperer/papers.html (Dec. 2001).

from adoption) and increases the incentive to adopt (i.e., increases their *marginal utility*). The software market is characterized by both "direct" and "indirect" network effects.[45]

- *Direct network effects* arise if my utility from using a good is directly increased when other people use the same good, as well. For example, if I want to share files with other people, it is important to me that my software is compatible with other people's software.
- *Indirect network effects* arise if wider adoption benefits users by changing the behavior of third actors, e.g. sellers of that good, or sellers or buyers of some related goods. For example, if an operating system is widely adopted, then other software companies who write application software have a stronger incentive to develop software that is compatible with this operating system.

Network effects often give rise to *external effects*.[46] If customer A decides whether or not to adopt a certain software product, he does not internalize the effect his adoption decision has on other customers. This suggests that there will be *under-adoption* of the network good. Even if priced at marginal cost, too few consumers will buy the network good because they do not take into account the positive external effects of their adoption on other consumers.

We will say that network effects are *strong* if they outweigh preferences for product A vs. product B, i.e. a consumer wants to adopt A if all other consumers adopt A, even if this consumer would have preferred that everybody adopts software B.[47] In this case "all adopt A" and "all adopt B" are both Nash equilibria of a coordination game, and efficiency requires that the entire market adopts the same product. We will say that network effects are *weak* if for at least two products there exist groups of customers who prefer one product no matter what all other consumers adopt.

With strong network effects there may be coordination problems. If consumers have different expectations about which software product will be the standard in the future, several incompatible products may coexist on the market for an extended period of time, which may be very inefficient. For example, a potentially serious problem of OSS is "forking." If software developers disagree on how a certain soft-

---

45. There may also be pecuniary network effects, e.g., if the fact that more people adopt a certain good reduces its price due to reduced production costs or improved terms of trade for consumers.

46. Note that not all network effects are external effects. First of all, network effects may be pecuniary. Secondly, it may be possible to internalize network effects through contractual provisions. *See* Farrell & Klemperer, supra note 44, at 44–45.

47. *See id.* at 28.

ware product should be developed, it sometimes happens that different groups of software developers disseminate different versions that are incompatible with each other. A proprietary software developer who owns and controls his software can easily prevent this from happening by imposing a development path that must be followed. In contrast, OSS is not controlled by anybody, so it is much more difficult to prevent forking. The open-source movement recognizes this problem and tries to solve it by imposing a social norm against forking.[48]

Even if consumers manage to coordinate their expectations, they may still fail to coordinate on the most efficient network good. For example, if there are two competing products, then there is an equilibrium if all consumers buy software A, but there is also an equilibrium if all consumers buy software B. Which product is adopted depends on which good consumers *expect* to be adopted. Even if everyone agrees in software A's superiority, an expectation that everyone will adopt software B will be self-fulfilling. Thus, with network goods there may be a market failure if consumers coordinate on the "wrong" software.

However, there are several arguments suggesting that consumers may be able to coordinate efficiently. Let us first consider the case where the network goods can be Pareto-ranked, i.e. all consumers agree which good offers the highest surplus.

- If consumers' adoption decisions are sequential, a simple backward induction argument suggests that they will coordinate on the Pareto-efficient network good. This is most easily understood in the case of just two consumers. The second consumer will adopt whatever the first consumer bought. Anticipating this, the first consumer will buy the network good offering the highest surplus. This argument is particularly convincing if there is one large customer — or a large group of customers coordinating their behavior — who moves first.
- If consumers can coordinate their behavior through communications or contracts, they will coordinate on the good that offers the highest surplus.
- If expectations track surplus, i.e. if each consumer expects that all other consumers will adopt the good that offers the highest social surplus, then everybody will adopt the Pareto superior good.

---

48. Raymond, *supra* note 23, discusses at length why forking could be a serious problem of the open source movement and why there is a "taboo" against it: "There is strong social pressure against forking projects. It does not happen except under plea of dire necessity, with much public self-justification, and with a renaming." *Id.*

But it is also possible that consumers coordinate on the wrong good, in particular if there is inertia in adoption. For example, one possibility is that after good A has been widely adopted a new good B is developed that offers a higher surplus on a clean-slate comparison. But if everybody expects everyone else to stick to good A, nobody is going to adopt the superior good B. On the other hand, if there is a large installed base of A and if consumers did make relationship specific investments that are lost if they switch to B (see the discussion of switching costs below), it may be more efficient that customers stick to A rather than switching to B.

If the network goods are differentiated and if some consumers prefer that everybody uses software A while others prefer product B, the analysis gets a little more complicated. Suppose again that network effects are strong in the sense that it is efficient that everybody adopts network good A. If expectations track surplus in the sense that people expect that eventually the most efficient network good will prevail, or if a critical mass of consumers who prefer software A can coordinate their behavior, then software A will again be adopted. However, if early adopters prefer good B, later adopters may be induced to buy good B as well, even if this is socially inefficient. But, again, switching costs of early adopters have to be taken into account when the efficiency of moving to good A is assessed.

Thus, strong network effects make markets "tippy." One network will tend to acquire an overwhelming market share. If consumers manage to coordinate on the right network, this monopolistic outcome is socially efficient. However, which network wins may depend on history and be influenced by large early movers, such as the government or a group of customers who coordinate their behavior by setting standards.

If network effects are "weak" so that two or more networks can coexist in the market, for example because the goods are strongly horizontally differentiated, then network externalities are much smaller. If network A wins an additional customer from network B, network A has a positive external effect on all members of network A but a negative external effect on all users of network B. The sign of the net effect depends on the relative sizes of the two networks. If network A is small as compared to network B, then there are relatively few people who benefit from the positive externality of the customer who switched to A, while there are many people who suffer in network B, so that the net effect is likely to be negative. Two networks can also coexist in the market if they are compatible with one another. In this case, the conventional lesson applies that market power depends on the degree of product differentiation.

The discussion so far completely ignores pricing. The large body of literature on pricing with network effects does not offer a clear

conclusion.[49] Network effects may weaken price competition, but they may also intensify price competition as compared to a situation without network effects, because each firm wants to acquire a critical mass of consumers that tips the balance in its favor.

2. Switching Costs

A product has *switching costs* if a buyer will purchase it repeatedly or if he will purchase complementary goods from the same seller in the future. The switching cost arises because the consumer has made an investment specific to the seller that would have to be duplicated if he switches to a different seller.[50] This investment may be in equipment, in learning how to use the software, in buying complementary software that is not compatible with the other product, or in setting up a relationship with the seller.[51] After consumers are locked-in with one seller, this seller may have an incentive to exploit his ex-post monopoly power. However, this does not necessarily imply a market failure.

- If firms can commit to future prices and qualities, competition will be for the "lifecycle" of purchases. Each consumer looks at which seller offers the highest net surplus from the bundle of goods she wants to consume — now and in the future — and prices will be driven down to the competitive level.
- Even if such complete contracts on future prices and qualities are not feasible, the outcome may still be highly competitive. To see this, suppose that there are two firms each offering two complementary goods, e.g. an operating system and application software and that a consumer wants to buy the two goods in sequence. If she bought the OS from firm A, she would have to incur a switching cost if using the application software of firm B. Thus, firm A will charge a price for the application software that exploits the switching cost of this consumer and earn a positive rent in the application software market. However, the prospect of this rent will intensify competition for the first good and drive its price below cost. This price pattern of bargains followed by rip-offs is similar

---

49. See *supra* note 46 for a survey.

50. *See id.* at 8.

51. In the following we will assume that switching costs are real social costs. Switching costs could also be contractual or purely pecuniary, e.g., in airlines' "frequent-flyer" programs or in "loyalty contracts" that offer rebates for repeated purchases.

to the price patterns of other goods with measurable switch-
ing costs.[52]

Nevertheless, switching costs may give rise to inefficiencies. In
particular, if sellers cannot price discriminate perfectly, there may be
socially excessive consumption of the first good that is priced below
cost and too little consumption of the second good that is priced above
cost. Furthermore, there may be inefficient switching. If a consumer
constitutes a large fraction of the market, it may be profitable to use
goods from both sellers and to incur the switching cost in order to
affect the terms of trade for future goods. While this may be a profit-
able strategy in markets such as the defense industry, where there are
few buyers with substantial market power, it does not seem to make
much sense in the market for packaged software. Even the software
purchases of a big company or of the government of a large country
are small as compared to total market demand. Finally, switching is
inefficient because relationship-specific investments have to be dupli-
cated.

The main concern about switching costs is their effect on market
entry. It is often claimed that high switching costs make it too difficult
for newcomers to enter the market but the effect is ambiguous.[53]

- If there is a constant flow of new customers, switching costs
  may make small-scale entry too easy. Because the incumbent
  firms have a stock of customers who are locked in already,
  they are not willing to lower their prices in order to compete
  for new customers. Therefore, a new entrant with no cus-
  tomer base can enter the market and sell to new customers
  even if this entrant has higher costs than the incumbent, re-
  sulting in excessive entry from a social welfare point of view.
- On the other hand, if entry has to be large scale (e.g., because
  of large economies of scale or due to network effects) it may
  be very difficult to take over the entire market. However, this
  does not imply a market failure. Even if the entrant can pro-
  duce the good at a lower total cost than the incumbent,
  switching is inefficient if the total cost saving is smaller than
  the switching costs that would have to be incurred by all ex-
  isting customers. If switching costs are large and there are
  real social costs, it is efficient that there is no market entry.
- Entry may also be too hard if switching costs are created arti-
  ficially by the incumbent, for example, if the switching costs
  are contractual — as with "loyalty" programs — or if the in-

---

52. A well-known example is the strategy to give away free razors in order to sell more
razor blades.

53. *See* Farrell & Klemperer, *supra* note 44, at 29–32.

compatibility between different products is not warranted for technological reasons but imposed by the incumbent to deter potential entrants.

- Entry may also be too easy or too hard for other reasons. For example, it may be too hard if entry reduces the market price so that some of the surplus from entry is captured by consumers, or it may be too easy if its main effect is to shift profits from the incumbent to the entrant. However, these arguments have nothing to do with switching costs and they are not specific to the software market.

To summarize, if there are significant switching costs it is likely that locked-in consumers are charged prices above costs by a proprietary software developer, there is little actual switching taking place, and large-scale entry is difficult and rare. However, this does not imply that markets with switching costs are uncompetitive or that there is a market failure. As we have shown, competition in switching-cost markets can be intense and need not generate supra-normal profits.

The general conclusion is that the effect of network effects and switching costs on the market for software is largely independent of whether the software is proprietary or open source. Network effects tend to favor market concentration. However, in most software markets at least two products maintain significant and stable market shares over extended periods of time, either because the two products are largely compatible with each other or because consumer preferences are sufficiently differentiated so that for most software markets, network effects seem to be weak. There may be some software products with strong network effects in the sense that it is socially efficient that a single product dominates the market. These markets are "tippy" — the software product adopted may depend on what large early movers are doing. Even if, at a later stage, a new software is developed that is technologically superior on a clean-slate comparison, it may be socially inefficient to switch to the new software if there is a large installed base and if switching costs are high.

One could argue that heightened antitrust scrutiny is warranted if firms choose to make their products incompatible, or if they create switching costs by imposing appropriate contracts on customers or through product design, even though this is not technologically required. But, even in this case, it is not clear that the government should impose compatibility standards or force a firm to lower switching costs because this may expropriate the incumbent's *ex ante* investments and discourage future innovations.

IV. Scope for Government Intervention and Possible Distortions

In recent years there have been many political initiatives trying to foster the open-source movement and to spread the use of OSS in public administration and at schools and universities. For example, in a German initiative called "Bundestux," members of parliament from all major political parties declared that the "introduction of a free operation system in the Bundestag" would be necessary to promote "basic regulation, competition and location policy, as well as for democratic reasons."[54] In France, three French senators presented a draft bill to the Senate, seeking a ban on the use of proprietary software in government departments. The proposal was defeated, but in August 2001 French prime minister Lionel Jospin handed down a decree creating the Agency for Technologies of Information and Communication in Administration, whose mission is "to encourage administrations to use free software and open standards."[55] The U.S. government has provided substantial support for R&D efforts that create OSS that must be released under the GPL.[56] Several Italian municipalities, including Florence and Pavia, have passed motions mandating the use of "software libero."[57] The government of Peru plans a bill that "makes it compulsory for all public bodies to use only free software."[58] The Taiwanese government has just passed a "National Open Source Plan" that requires all schools and public agencies to switch to OSS within the next three years.[59] In Norway, Statskonsult (the Norwegian directorate on public management) has prepared a report on the usability of Linux in the Norwegian public sector. The main conclusion is that the government should support the development of OSS to promote alternatives to current software and that it should encourage schools to adopt Linux.[60] Finally, the European

---

54. *See* Robert W. Smith & Stefan Krempl, *Tux Takes Its Seat in Germany's Federal Parliament*, HEISE ONLINE *at* http://www.heise.de/english/newsticker/data/anw-28.02.02–006/ (Feb. 28, 2002).

55. *See* Paul Festa, *Nations Uniting for Open Source*, *at* http://news.zdnet.co.uk/story/0,,t269-s2094089,00.html (Aug. 28, 2001).

56. See Thomas Sterling, *Beowulf Linux Clusters*, *at* http://beowulf.gsfc.nasa.gov/tron1.html (last visited Jan 5, 2003); and Evans & Reddy, *supra* note 6, § VI(D), for several other examples.

57. *See* Festa, *supra* note 55.

58. *Peru: Gesetz soll Nutzung freier Software vorschreiben*, *at* http://www.golem.de/0205/19653.html (May 6, 2002).

59. *Der Staat macht sich für Linux stark; Das offene Computer-Betriebssystem, das mit der Microsoft-Software konkurriert, gewinnt an Bedeutung; Innenminister Schily schließt Kooperationsvertrag mit IBM*, SUEDDEUTSCHE ZEITUNG, June 5, 2002, at 26.

60. *See Rapport 2001:7 Åpen programvare*, *at* http://www.statskonsult.no/publik/publikasjoner/2001–07/r2001–07eng.pdf (July 2001), for an English translation of the report.

Commission has recently published an extensive study on the use OSS in the public sector.[61]

While there is a lot of public support for these policies, they are not uncontroversial. Free-market advocates distrust any government intervention. They argue that the software market, like any other market, should not be interfered with, except possibly by antitrust policy.[62] Some proponents of the open-source movement, in particular those with a libertarian background, come to the same conclusion, deeply distrusting any public intervention by bureaucrats and politicians and believing that the OSS "bazaar"[63] will succeed on its own merit. They argue that public policy should be limited to promoting compatibility standards that serve public needs rather than specific groups or corporate interests.[64]

In this section we will discuss the economic merits of various policies of the government that are being suggested to support OSS.

A. Direct Subsidies for Specific Open-Source Projects

The government could directly support a particular open-source project either by offering direct subsidies to the project or by employing computer experts at universities or government agencies to support it. In all developed countries, a large fraction of total R&D expenditures is paid for by the government.[65] However, the government should restrict itself to subsidizing basic research. Basic research is a public good with strong positive external effects that will not be provided by the market. Typically, it is unclear what the results of basic research are going to be, how long it will take to find them, and what they may eventually be used for. The potential positive spillover effects of basic research are widespread and very difficult to internalize by commercial companies, therefore they have little financial incentive to engage in it. This is why basic research has to be carried out at universities or publicly-financed research labs.

---

61. *See* PATRICE-EMMANUEL SCHMITZ & SÉBASTIEN CASTIAUX, EUR. COMM'N, POOLING OPEN SOURCE SOFTWARE, *at* http://europa.eu.int/ISPO/ida/export/files/en/1115.pdf (June 2002).

62. *See, e.g.*, Jeff Taylor's comments in the round table discussion issued by *The American Prospect at* http://www.prospect.org/controversy/open_source/.

63. ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR 21–63 (rev. ed. 2001), *available at* http://www.firstmonday.org/issues/issue3_3/raymond/index.html (arguing that the very decentralized, bazaar-type, bottom-up governance structure of the open source movement is superior to the centrally organized, top-down governance structure of large commercial software companies).

64. *See, e.g.*, Eric Raymond's comments in the round table discussion issued by *The American Prospect*, *supra* note 62.

65. For example, MICHAEL L. KATZ & JAMES A. ORDOVER, R&D COOPERATION AND COMPETITION, 137–91 (Brookings Papers of Economic Activity, 1990), report a 47 percent subsidy rate for private sector research in the U.S. in 1988.

The development of most software products, however, is applied R&D for which these problems are much less severe. Applied R&D is directed to the development of a specific product with certain characteristics or the solution of a well-defined technological problem. Here, the outcome of the R&D process is more predictable, making it easier to internalize positive spillover effects by seeking patent or copyright protection and by licensing the newly developed technology to third parties. Therefore, applied R&D can be provided by the market.

Furthermore, for applied R&D it is very important that the product fulfills the needs of potential customers. A profit-maximizing firm has a strong incentive to do so. As a product is adapted to meet the needs of its consumers, consumer interest will increase, as will the price that can be charged for product. Thus, a commercial company will focus on what customers want, and it will try to do so in the most cost-effective way. A firm will only be able to survive on the market if it develops a better product at lower cost than its competitors. In contrast, a government-sponsored research lab does not face the constraints of the market and has much less incentive to focus on customer needs and cost efficiency.

Another problem with public subsidies to applied R&D is that they invite rent-seeking activities. A project that wants public funding does not have to beat competing projects on the market, it has to beat them by lobbying for stronger political support. Researchers typically do not like to do this. In order for their projects to survive, they sometimes accept the support of large commercial companies who may have a somewhat different agenda. There are many examples of well-intended scientific projects that got captured by large companies who then managed to acquire vast amounts of public subsidies (e.g., in the space, defense and nuclear industries).[66]

While the arguments given so far apply to all industries, there are some additional important arguments that stem from the specific characteristics of the software market. We have seen in Section III that there may be strong network externalities in software markets that make markets "tippy." Because of these network externalities, one product will capture the lion's share of the market. If consumers coordinate on the right product, this outcome is also efficient. However, the product that succeeds on the expectations of consumers, which may in turn be affected by the behavior of large players such as the government. Therefore, even modest subsidies to one particular project may have a strong impact on the market outcome. In the extreme, government support for one project could make the market tip to another equilibrium in which the government-sponsored product drives its competitors out of the market. Governments have a poor track re-

---

66. *See, e.g.*, JAMES Q. WILSON, BUREAUCRACY (1989).

cord in picking winners through "industrial policies."[67] The government has neither the ability nor the right incentives to determine which software product is the most efficient. This decision should be left to the market, and the government should restrict itself to providing a level playing field.

Finally, if the government decides to subsidize some basic research in software development, it should ensure that this research is broadly disseminated and that it can be used by everybody. Thus, publicly-sponsored software should be put in the public domain or protected by liberal licenses such as BSD. Support for projects that are licensed under the GPL is unsuitable in this respect. Because of the viral nature of the GPL, proprietary software cannot use GPL software without turning itself into GPL software. This encourages the development of two incompatible networks with significant welfare losses for consumers.

B. Adopting OSS in the Public Sector

Several governments are currently considering proposals prohibiting government agencies from using proprietary software if an alternative open-source product exists, and forcing schools and universities to switch to open source.[68]

The proponents of this policy argue that OSS is qualitatively better than proprietary software and that the total cost of ownership, including the costs of maintenance and technical support, is lower. If this is the case, which may well be possible for some software applications, then government agencies as well as private firms and consumers should be happy to switch to open-source products, and no coercive actions by the government are needed to induce them to do so.

However, in some cases open source does not seem to be the best and most cost-effective solution. For example, the German Bundestag ordered a study from an independent consulting firm, INFORA (based in Berlin), to compare five different software solutions, ranging from pure open source to pure proprietary software, for the computer infrastructure used by all members of the Bundestag. INFORA recommended a solution that is largely based on proprietary software and uses Linux only as an e-mail server and Groupware solution. It argued that open-source solutions are yet insufficient for the requirements of

---

67. *See, e.g.*, Scott J. Wallsten, *The Effects of Government-Industry R&D Programs on Private R&D: The Case of the Small Business Innovation Research Program*, RAND J. ECON. 82-100 (2000); Linda R. Cohen & Roger G. Noll, *Feasibility of Effective Public-Private R&D Agreements*, INT'L J. ECON. BUS. 223-40 (1995).

68. See the examples given at the beginning of Section IV.

the parliamentarians.[69] Nevertheless, after a heated debate accompanied by intense lobbying activities from both proprietary software developers and the open-source movement, the Bundestag decided that the solution for the 150 servers of the Bundestag would be Linux-based while the 5000 desktop computers will initially feature Windows XP. This solution is considerably more expensive than the one suggested by INFORA and cannot be justified on the merits of OSS alone.[70]

Therefore, proponents of a move to open source sometimes rely on the additional argument that OSS should be adopted in order to strengthen a second source that puts competitive pressure on the incumbent proprietary software developer. It is argued that this is desirable in order to force the incumbent to lower her prices.

However, while it is clearly true that the existence and rapid development of the open-source movement puts competitive pressure on proprietary software developers and constrains their pricing behavior, this does not imply that the government should intervene in the market by favoring OSS.

1. Public Subsidies for OSS with Strong Network Effects

Let us first consider the case where network effects are strong in the sense that only one product will capture the lion's share of the market. In this case, the government could make the market tip. For example, if it forces schools and universities to adopt OSS, it will significantly lower the costs of teaching students how to work with OSS. If there are high switching costs involved in moving to proprietary software afterwards, OSS may capture the entire market even if it is not qualitatively superior to proprietary software. Thus, it may be the government that picks the winner. Second, if a significant part of the market is directed to open source, the current and future profits of proprietary software companies are significantly reduced. Therefore, they have lower incentives to innovate. Third, this policy may discourage the entry of other proprietary software developers if they feel that they cannot compete with the OSS that is favored by the government. Finally, if the OSS favored by the government is licensed by the GPL or other viral licenses, which make it legally difficult for proprietary software developers to make their software compatible with OSS, then the government will foster a development in which there are two incompatible networks on the market. This is inefficient, be-

---

69. It is estimated to cost an additional Euro 80,000 per year. *See Open Source im Bundestag als `strategischer Vorteil',* HEISE ONLINE *at* http://www.heise.de/ newsticker/data/odi-27.02.02–000 (Feb. 27, 2002); Smith & Krempl, *supra* note 54.

70. *Id*.

cause significant positive network effects are lost, and it may reduce competition in the market.

2. Public Subsidies for OSS when Network Effects Are Weak

The less extreme and probably more realistic case is where network effects are weak and/or software products are largely compatible with each other so that two or more products can survive on the market in the long run. Even in this case, a move to OSS by the government is not necessarily going to increase competition or lower prices. In fact, under natural circumstances the exact opposite will happen.

Suppose that there are two different software products offered on the market. One is proprietary software that is sold by a profit-maximizing, proprietary software developer, while the other is OSS that is distributed for free. We can distinguish three groups of consumers on this market. Some consumers will always buy the proprietary software because they are unsophisticated users who are unable to use the OSS product. Some consumers will always go for the OSS because they want to adopt the software to their specific needs. Some consumers are considering both types of software and base their decision on the respective qualities, the price of the proprietary software, and their idiosyncratic preferences.

Suppose now that the government forces some government agencies, belonging to the third group of consumers, to adopt the OSS product. This reduces the market size of the consumers for whom there is competition. The proprietary software developer then has less of an incentive to reduce his price in order to compete for these customers. Rather, the developer will raise his price in order to make a larger profit on the first group of consumers. Thus, the price for the proprietary software is going to rise. As a consequence, consumers, government agencies, and the proprietary software developer are worse off. Consumers have to pay higher prices for the proprietary software while the price for OSS is unchanged. Government agencies do not have a choice but are forced to adopt OSS, and the proprietary software developer loses part of his profits. On the other hand, the OSS developer does not benefit in financial terms from this policy because she continues to offer her product at a price of zero.

Let us now consider the incentives of software developers to invest in product improvements. The incentives to innovate for the proprietary software developer are reduced because the size of the market of undecided consumers, in which there is quality competition, is reduced. Thus, a quality improvement will induce fewer consumers to switch to the proprietary software, and it is less profitable for the proprietary software developer to compete on the basis of quality. On the other hand, the incentives of the OSS developer are not directly af-

fected. Her market size has increased, but she does not benefit financially from quality improvements in any way.

To conclude, if network effects are weak and if there is competition on a market for a particular software product, government intervention that fosters the use of open source is likely to reduce competition, increase prices, reduce the incentives to innovate, and make decrease net social welfare. In the Appendix, we present a sample model.

C. Subsidies for Institutions that Coordinate Open-Source Development

In the alternative, governments could subsidize institutions of the open-source movement that try to coordinate software development and standard setting. An example is BerliOS, a mediator for open-source developers and customers, which is co-funded by the German federal government and private companies such as Hewlett-Packard and Linux Information Systems.[71] This policy may have some merit, in particular if the role of such an institution is neutral towards any particular open-source project and restricted to encouraging open standards and compatibility with open-source and proprietary software. Thus, as argued above, the government should not promote open-source projects that are licensed by the GPL, which makes this software incompatible with proprietary and other OSS.

However, the basic problems of any government intervention remain. It is not the job of the government to decide which software is going to be the standard of the future. Furthermore, any financial support to particular projects is going to invite rent-seeking activities, and political pressures will bias the decision regarding which projects will be promoted.

V. Conclusion

In this paper, we have shown that OSS and proprietary software do have different impacts on the market outcome. OSS is priced at marginal cost, which is efficient from a static point of view. However, innovation and technological progress have been extremely important for the software market since its very beginnings, and this is likely to continue. From a dynamic perspective, marginal cost pricing is inefficient, because it gives insufficient incentives for software developers

---

71. "The goal of BerliOS is to provide support for different interest groups in the area of Open Source Software (OSS). Our aim is to fulfil [sic] a neutral mediator function. The target groups of BerliOS are on one hand the developers and users of Open Source Software and on the other hand commercial manufacturers of OSS operating systems and applications as well as support companies." *See* "BerliOS," *at* http://www.berlios.de/index.php.en (last visited Feb. 11, 2003).

to engage in R&D. There are significant incentives for professional programmers to contribute to OSS, in particular to fix bugs and to adopt software meeting their particular needs. However, the social benefits of a new and innovative software product are far greater than the modest private benefits that an open-source contributor reaps. Furthermore, a proprietary software developer has a strong financial incentive to write software that is as closely adapted to the needs of *all* consumers, while OSS developers tend to be more responsive to the specific needs of sophisticated users and IT professionals.

Therefore, despite the impressive record of OSS development in recent years, it cannot fully replace proprietary software. If we want software development to continue to flourish, and if we want better and more sophisticated software solutions to be developed for new socially-useful applications, then the profit motive is crucially important to spur this innovation. The profit motive aligns the interests of the software developer and the interests of society. In order to be able to make high profits, the proprietary software company must make its software as useful as possible to as many consumers as possible. As Adam Smith put it more than two centuries ago:

> "It is not from the benevolence of the butcher, the brewer, or the baker that we expect our dinner, but from their regard to their own interest. We address ourselves, not to their humanity but to their self-love, and never talk to them of our own necessities but of their advantages."[72]

This powerful incentive is largely absent for OSS developers. Nevertheless, OSS is an important complement to proprietary software. For some software products the open-source solution seems to be qualitatively better or at least more cost-efficient than the proprietary software solution, and on some software markets, OSS already has a dominant market share by some measures (e.g., the Apache web server or Sendmail).[73] For some other products, it is still unclear what the superior solution will be. For these products, open source is putting competitive pressure on proprietary software developers. This is beneficial because it reduces prices and spurs competition in quality and innovation.[74]

---

72. ADAM SMITH, AN INQUIRY INTO THE NATURE AND CAUSES OF THE WEALTH OF NATIONS 17 (Edwin Cannan, ed., University of Chicago Press 1976) (1776).

73. *See, e.g.*, Evans, *supra* note 2, at 35.

74. Many empirical economists claim that there is an inverted U-shaped relationship between the degree of competition, as measured by the number of competitors, and the incentives to innovate. *See* FREDERIN M. SCHERER & DAVID ROSS, INDUSTRIAL MARKET STRUCTURE AND ECONOMIC PERFORMANCE 646–651 (3d ed. 1990), for a survey. Thus, incentives are strongest if there are just a few firms who fiercely compete for the market.

However, the government should not interfere with the market and artificially favor any specific product or OSS as a whole. If there are strong network effects, the market equilibrium depends on consumers' expectations regarding future adoption decisions of other players. In this case, government intervention may make the market tip in one direction. Like in any other market, it is not the job of the government to pick winners and losers. Furthermore, if network effects are less strong and if two competing products are going to stay in the market, artificially favoring one product by requiring government agencies to buy it may actually reduce competition, increase prices, and lower innovation and social welfare.

Thus the government should restrict itself to subsidizing basic research in software technology at universities and other academic institutions in order to promote new scientific developments that could be the basis for new products, whether proprietary or open source, and to facilitate standard setting and encourage compatibility.

See Klaus M. Schmidt, *Managerial Incentives and Product Market Competition*, 64 Review of Economic Studies, 191–214 (1997), for a model of managerial incentives to innovate and to reduce costs that captures this effect.

APPENDIX: A SIMPLE MODEL OF GOVERNMENT ADOPTION OF
OSS IN A SOFTWARE MARKET WITH HORIZONTAL PRODUCT
DIFFERENTIATION

Consider a software market in which two software products are offered, proprietary software (PS) and OSS. The proprietary software producer maximizes profits, while the OSS developer sells at marginal cost, which is normalized to 0. We will assume that the size of the market is fixed and that the two software products are compatible with each other, so that we can ignore network externalities. There are three groups of customers. Some customers (of mass N_p) will always buy the proprietary software, some (of mass N_o) will always buy the OSS, and some customers (of mass N_u) may buy either of the two products. We model the preferences of these last customers with a simple Hotelling model of horizontal product differentiation. They are uniformly distributed on the unit interval. A consumer at location $x \in [0,1]$ gets a net utility from buying the proprietary software of

$$U_u(PS) = v_p - t \cdot x - p \qquad (1)$$

where v_p reflects her gross utility from using the software, $t \cdot x > 0$ is her "transportation cost" (e.g., her cost of learning how to use this software or the cost of adapting other software applications), and p is the price that she has to pay for the proprietary software. If this consumer buys the OSS product, her net utility is given by

$$U_u(OSS) = v_o - t \cdot (1-x) \qquad (2)$$

where v_0 is her gross utility from the OSS which may be larger or smaller than v_p. Note that OSS is priced at 0.

Consumers differ only in their location, x. In the following, we will assume that both software developers serve some part of the undecided customers in equilibrium.[75] Thus, the marginal consumer, $\overline{x}$, who is just indifferent between buying PS or OSS is characterized by

---

75. This is the case if it is neither optimal for the proprietary software developer to focus just on customer group Np, requiring that

$$v_p N_p < \frac{[N_u(v_p - v_o + t) + 2tN_p]^2}{8tN_u}$$

nor is it optimal for the proprietary software developer to capture all of the undecided customers, requiring that

$$(v_p - v_o - t)(N_p + N_u) < \frac{[N_u(v_p - v_o + t) + 2tN_p]^2}{8tN_u}$$

$$v_p - t\bar{x} - p = v_o - t(1 - \bar{x}) \qquad (3)$$

which yields

$$\bar{x} = \frac{v_p - v_o + t - p}{2t} \qquad (4)$$

Thus, the proprietary software developer maximizes

$$\Pi_p = p[N_p + \bar{x}N_u] \qquad (5)$$

Substituting (4), the first order condition for the profit maximizing p^* yields

$$p^* = \frac{v_p - v_o + t}{2} + \frac{tN_p}{N_u} \qquad (6)$$

Substituting (6) in (4) and in (5) we get

$$\bar{x} = \frac{N_u(v_p - v_o + t) - 2tN_p}{4tN_u} \qquad (7)$$

and

$$\Pi_p^* = \frac{[N_u(v_p - v_o + t) + 2tN_p]^2}{8tN_u} \qquad (8)$$

Suppose now that the government decides to force some agencies or schools and universities to adopt the OSS, even though they may have preferred to buy the proprietary software. We can model this as an increase of N_o at the expense of N_u, i.e. $\Delta N_o = -\Delta N_u > 0$. The effects of this policy are described in the following proposition:

> **Proposition 1:** If $\Delta N_o = -\Delta N_u > 0$, then the market share of proprietary software is reduced and the price for proprietary software increases. The welfare effect of this policy is unambiguously negative, i.e. all players (consumers, producers, and the government) are worse off.

The proof of the proposition and its intuition are straightforward. Differentiating (6) and (7)
with respect to N_u yields

$$\frac{dp^*}{dN_u} = \frac{-tN_p}{N_u^2} < 0 \qquad (9)$$

and

Both conditions are satisfied if, *e.g.*, *Np* is not too large as compared to *Nu* and *vp-vo* is not too large as compared to *t*.

$$\frac{d\bar{x}}{dN_u} = \frac{(v_p - v_o + t)4tN_u - 4t[N_u(v_p - v_o + t) - 2tN_p]}{8t^2 N_u^2} > 0$$

(10)

Thus, if the market of undecided consumers shrinks, the proprietary software developer will focus more on the customer group N_p and the competition for the undecided consumers is reduced. Therefore, he will raise prices and lose market share.

The proprietary software developer is worse off after the policy change. To see this formally, note that

$$\frac{d\Pi}{dN_u} = \frac{2[(v_p - v_o + t)N_u + 2tN_p](v_p - v_o + t)8tN_u - 8t[N_u(v_p - v_o + t) + 2tN_p]^2}{[8tN_u]^2}.$$

(11)

This term is positive if and only if

$$N_u(v_p - v_o + t) > 2tN_p \quad (12)$$

which is implied by $\bar{x} > 0$. Consumers also suffer because they face higher prices. Finally, the government is worse off, because some of its agencies have been forced to adopt OSS even though they would have preferred to buy proprietary software. Furthermore, all other government agencies also suffer from higher prices and less competition in the market for undecided customers.[76]

Let us now consider how the incentives to innovate are affected by this policy. Suppose that before the market game described above is played, there is a stage 0 at which the proprietary software developer can increase v_p by y if he invests $c(y)$ in R&D, where $c(y)$ is strictly increasing and convex in y. Thus, his profit in the market game is given by

$$\Pi_p^*(y) = \frac{[N_u(v_p + y - v_o + t) + 2tN_p]^2}{8tN_u} - c(y). \quad (13)$$

Differentiating with respect to y and simplifying terms yield the following first-order condition for the profit-maximizing investment in R&D:

$$\frac{d\Pi_p^*(y)}{dy} = \frac{N_u(v_p + y - v_o + t) + 2tN_p}{4t} - c'(y) = 0.$$

(14)

---

76. The open-source software developers make zero profits before and after the policy change, so they are not affected. However, they may derive some non-monetary benefits from the wider adoption of their software product.

We are interested in the question of how the incentives to innovate are affected if the government increases N_o at the expense of N_u. Using the implicit function theorem, we get

$$\frac{dy}{dN_u} = -\frac{\dfrac{d(14)}{dN_u}}{\dfrac{d(14)}{dy}}. \qquad (15)$$

The denominator is negative which is implied by the second-order condition of the profit-maximization problem. The numerator is given by

$$\frac{d(14)}{dN_u} = \frac{v_p + y - v_o + t}{4t} > 0. \qquad (16)$$

Thus, we get

> **Proposition 2:** If the government reduces N_u by forcing some agencies to adopt OSS, the incentives of the proprietary software developer to improve the quality of his software are reduced which further reduces the welfare of all market participants.

Note that if the government decides to adopt OSS, this does not imply that open-source developers are going to innovate more or develop new OSS. There is no financial incentive to do so, because OSS is given away for free.

To conclude, this simple model demonstrates that forcing government agencies to adopt OSS need not increase competition and foster the development of new and better software, but it may have the exact opposite effect. Competition is reduced because the decision by a significant part of the market about which software to adopt is politically determined and thus independent of prices and qualities that are offered by the contestants.