Public Subsidies for Open Source?
Some Economic Policy Issues of the Software Market[1]

Klaus M. Schmidt[2]
University of Munich, CEPR and CESifo

Monika Schnitzer[3]
University of Munich, CEPR and CESifo

This version: November 2002

Abstract: This paper discusses the economic merits of direct or indirect governmental support for open source projects. Software markets differ from standard textbook markets in three important respects that may give rise to market failures: (i) large economies of scale, (ii) crucially important innovations, (iii) significant network effects and switching costs. We analyze the differences between proprietary software and open source software with respect to these market features and ask whether open source as an alternative to proprietary software can mitigate these problems. Then we discuss the implications of various forms of governmental support for open source.

JEL classification numbers: H41, O31, O38

Keywords: Software Market, Open Source, Public Goods, Innovation Incentives, Public Subsidies

---

[1] We would like to thank David Evans, Joachim Henkel, Justin Johnson, Bernard Reddy and participants at the conference "Open Source Software: Economics, Law and Policy" in Toulouse, June 20-21, 2002, for helpful comments and suggestions. Financial support by National Economic Research Associates (NERA) is gratefully acknowledged.
[2] Department of Economics, University of Munich, Ludwigstr. 28 (Rgb.), D-80539 Munich, Germany, Tel.: +49-89-2180 2250, email: klaus.schmidt@lrz.uni-muenchen.de .
[3] Department of Economics, University of Munich, Akademiestr. 1, D-80799 Munich, Germany, Tel.: +49-89-2180 2217, email: schnitzer@lrz.uni-muenchen.de .

I. Introduction

The open source movement has been very successful in developing software products such as Linux, Apache or 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, e.g. by paying direct subsidies to open source projects, by standardizing on open source software in government agencies, or by requiring schools and universities to replace proprietary software by open source software whenever this is possible.[4]

There is also a small but rapidly growing academic literature on open source. Most of this literature tries to understand the governance structure of open source projects.[5] 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, or Intel contribute significant capital investments to open source projects? What business models are likely to be successful in an open source market?

However, the question of the economic merits of direct or indirect public subsidies for open source projects has received very little attention in the literature so far.[6] This paper tries to fill this gap. We do not seek to develop new theoretical models to address this question. The idea is rather to use modern economic theory in order to analyze the specific aspects of the software market in general and of open source software in particular. This allows us to better understand the functioning of this market and to develop a framework in which the implications of governmental support for open source projects can be discussed systematically.

---

[4] See Section IV for several examples of such initiatives.

[5] The economic literature on open source includes Lerner and Tirole (2002), Johnson (2001), Bessen (2001), Harhoff et.al. (2000) and Evans (2001). Many additional working papers are collected at and can be downloaded from http://opensource.mit.edu .

[6] A notable exception is Evans and Reddy (2002). They focus on the problem that arises when open source software is licensed under the terms of the "General Public License" (GPL, see below) and offer a more comprehensive survey on the institutional features of open source software and the empirical findings on software markets, while our paper focuses on the economic arguments in favor or against public subsidies for open source software.

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 licenses. In Section III we discuss potential market 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 as an alternative to proprietary software can mitigate these problems. In Section IV we discuss the economic merits of various policies of direct or indirect governmental support for open source, some of which are already being implemented by several countries. Section V concludes.

II. What is Open Source?

The general idea of "open source" is that the source code of a software program should be made available to everybody at no extra charge and that everybody should have 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 to machine code, which is just a sequence of zeros and ones. This machine code is very difficult to read for humans, and it is also difficult and time consuming to re-translate it into source code. Therefore, open source requires that not just the machine code, but also the source code is made freely available. Given the ready availability of the source code for open source software, firms generally can charge only low prices for such software: any recipient of the source code (which must be made available with the software) can freely redistribute the software, driving prices down toward average distribution costs.

In contrast, a license for "proprietary software" is sold like any other good or service. Because the firm that develops the software wants to make a profit, it has to be able to protect its intellectual property rights. The creator of a software program can obtain a copyright and (in some countries) a patent which 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. When the source code is made available to other firms, it is typically licensed under very restrictive conditions.

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

While the open source movement shares some general principles, there are also important differences in motivations and objectives. For example, the "Free Software Foundation", whose founder and most prominent speaker is Richard Stallman, argues that "free software" is (like, e.g. "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." [7] 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".[8] Other participants in the open source movement have a more pragmatic point of view: "When programmers can read, redistribute, and modify the source code of a piece of software, the software evolves" and "this rapid evolutionary progress produces better software than the traditional closed model".[9] They want to make this case to the

[7] See http://www.gnu.org/philosophy/ (accessed on May 11, 2002).
[8] Richard Stallman, "Free Software: Freedom and Cooperation", speech delivered at NYU, New York, 29 May 2001, http://www.gnu.org/events/rms-nyu-2001-transcript.txt (accessed on May 11, 2002), answer to last question.
[9] Open Source Initiative, http://www.opensource.org/ (accessed on May 11, 2002)}

commercial world and are prepared to co-operate with proprietary software developers in order to foster software development.

If open source software was not protected by copyrights or patents, anybody could take the software, modify it, obtain a copyright for the modified version, and exclude others from using it. Therefore, open source software has to rely on the protection of intellectual property rights as well. One of the first licenses for open source software was developed by the Free Software Foundation (FSF), founded by Richard Stallman of the MIT. This "GNU/General Public License" (GPL) requires that if somebody wants to distribute software that is covered by the GPL, he has to make the source code of this software available and he cannot restrict users to make the software and its source code available to others.[10] Furthermore, users have to agree that all enhancements of the code - and even code that combines the open source software with some other, separately developed software, has to be licensed according to the terms of the GPL.[11] 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.

An example of a different and more liberal license is the Berkeley Software Distribution (BSD) license: A user who chooses to redistribute a BSD-licensed program (whether modified or not) must retain the copyright notices in the program code, but faces no other serious restrictions. In particular, software licensed under BSD is not viral and does not render proprietary software, that builds on or interacts with it, open source. There are many other licenses that have been developed to preclude the commercialization of open source software. The Open Source Initiative (OSI) publishes a list of the most frequently used licenses and offers a guide to the terms a software license should (and should not) contain in order to be considered open source.[12]

[10] The FSF calls this feature of the GPL (and similar licenses) "copyleft".
[11] 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." See http://www.fsf.org/copyleft/gpl.html . For a more detailed critical discussion of the GPL see Evans and Reddy (2002).
[12] http://www.opensource.org/licenses/index.html (accessed May 10, 2002)}

III. Potential Market Failures of 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.[13] Total surplus is maximized if the good is priced at marginal cost. If the minimum efficient scale is small as 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, so that 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, while 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 characterized by economies of scale, and the market for a given software product is a "natural monopoly".[14] Therefore, static efficiency requires that the whole market is served by just one firm, and that this firm charges prices equal to marginal costs.

From a static point of view, open source software has the advantage that it is distributed for free (or at a small fee reflecting distribution costs), which is socially efficient. A proprietary software developer cannot price at marginal cost. If she does not charge at least average costs, where average costs include a risk-adjusted normal return on capital, she 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 costs can be sustained. There are several reasons why a monopolist may not be able to achieve supra-normal profits.

---

[13] Strictly speaking, this measure is appropriate only if there are no income effects. However, Willig (1976) has shown that 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.

[14] Note that increasing returns to scale imply here that the cost function is "strictly subadditive" which is the formal requirement for a "natural monopoly" in the sense of Baumol, Panzar and Willig (1982, p. 17).

- Even if competition within the market is not possible, there may be competition for the market.[15] 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 being taken over by a competitor. Thus, the crucial question is how large the barriers to entry are in the software market. The fixed costs that are necessary to develop a software product are largely sunk, so they are a bygone for the monopolist in the event of entry by a new firm. Thus, the incumbent monopolist may price below average cost if entry occurs which may turn entry unprofitable and deter potential entrants. Second, there may be network effects (in particular if the products offered by the incumbent and by the entrant are not compatible) and switching costs on the side of consumers which may prevent them from adopting the entrant's product even if it is priced below the incumbent's software (see Section III.C below). On the other hand, rapid technological progress on the software market may facilitate entry. There are many examples in the software industry 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 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 those who have not yet bought it but who have a willingness to pay that is higher than her marginal cost. This implies that over time prices are going to fall until they reach marginal cost. If consumers anticipate that this is going to happen, they will wait in order to buy at lower future prices. The Coase Conjecture[16] suggests that this may force the monopolist to lower her price very rapidly and to sell at close to marginal cost almost instantaneously.

  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

[15] See Demsetz (1968), and Baumol, Panzar, and Willig (1982).
[16] See Coase (1972) and Gul, Sonnenschein and Wilson (1986).
[17] See Sobel (1991).

quickly, so 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 are not going to buy the new version if the cost of this new version exceeds the value added to the older version.

To summarize, there are several constraints on the pricing policy of a proprietary software developer which may induce her to charge prices significantly lower than monopoly prices. Nevertheless, open source software 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* (with rapid technological change) efficiency requires not only that the good is sold to all consumers whose willingness to pay is higher than the marginal cost of producing the good. It also requires that firms have the right incentives to innovate efficiently. In such a Schumpeterian world, innovation is risky because only those firms will be able to survive on the market who are more successful innovators than their competitors. Hence, commercial firms engage in costly innovative activities only if they can protect their intellectual property and can charge more than average cost in case of success in order to compensate for the losses in case of failure.

Clearly, the market for software changed very rapidly over the last twenty years. Spurred by the enormous technological progress in the production of computer hardware many new and ever more sophisticated software products have been developed. Many software products that dominated the market in the 1980s and early 1990s have been leap-frogged by new and better

[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.

products offered by different software companies 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, it is obvious that innovation and technological process play a crucial role for the development of the software market.

A profit maximizing firm would have an efficient incentive to innovate if the expected profit from the innovation just equals the expected net gain of society from this innovation. In reality, this is very unlikely to be the case. The literature on the incentives for R&D and innovation has pointed out three external effects that may distort innovation incentives:[19]

- 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 that is generated by the innovation. This is a positive externality of R&D that tends to reduce the incentive to innovate as compared to the first best.
- 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 the innovation incentives.
- Finally, the *business stealing effect* is that the introduction of a superior technology makes some existing products less attractive and therefore takes away some of the rents that were previously earned by the producers of these old products. This externality is negative. Note that there is a private but no social return from this rent-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 externality outweights both the consumer-surplus and the R&D externality, so that the incentives to innovate are too strong.[20] However, it is generally believed that the much typical situation is for the overall externality of innovations to be positive,

[19] See e.g. Romer (1991), Grossman and Helpman (1991), and Aghion and Howitt (1992).
[20] Aghion and Howitt (1992).

so that the incentives to innovate are too weak. 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 open source software? At first glance, it seems puzzling that open source software exists at all. After all, there are no direct pecuniary incentives to develop a piece of software that is afterwards distributed for free to everybody who wants to use it. Some commentators have argued that the open source community is a "gift culture"[21] that is motivated by altruism and reciprocity. According to this view, people contribute to the public good of open source because they enjoy being part of the open source community and because they want to help others and to reciprocate 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.[22] Furthermore, while it is now widely acknowledged that reciprocity does play an important role for the interaction of people in small groups[23] it is very unlikely that altruism and/or reciprocity provide sufficient incentives to explain the enormous all, we do not observe similar patterns of behavior in most other areas of economic activity. So what additional incentives motivate programmers to write open source software?

Many contributors to OSS are sophisticated users. For them, the cost of fixing a bug, customizing a given software to their own needs, or developing a modest new application is often fairly small. Most of them learned UNIX (from which many open source programs are derived) at high school or university, so they do not have to invest in learning a new program. If they have access to the source code, it is relatively easy to spot weak points and to write new code for improvements. Furthermore, these improvements yield immediate benefits for their daily work. 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 to other people. 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?[24]

---

[21] See Raymond (2000b).
[22] See Lerner and Tirole (2002, Footnote 13).
[23] See e.g. Fehr and Schmidt (2002) for a survey on the recent theoretical and experimental literature on this topic.
[24] See Johnson (2001) 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 von Hippel (1988) and Harhoff et.al. (2000) 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

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 completely new piece of software. 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.[25] Lerner and Tirole (2002) argue that there are strong signaling incentives that can explain this behavior. 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.[26] 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.[27]

This 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. Raymond points out that "removing a person's name from a project history, credits or maintainer list is absolutely not done without the person's explicit consent. ... Surreptitiously filing someone's name off a project is, in cultural context, one of the ultimate crimes."[28] Second, most open source programs have been 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 there is far less mass market application software written by the open source movement. Third, most open source projects have a modular structure. This is not only necessary in order to facilitate the cooperation of many independent programmers who may be

---

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. Furthermore, by freely revealing an innovation, a user may induce other users to adopt this solution as well which may set a new "standard" which in turn induces other manufacturers to provide complementary product improvements. These incentives are more likely to work if their is no competition between users.

[25] Mockus et.al. (2000) report that 83% to 91% of changes in the Apache code have been written by just 15 top developers.

[26] These delayed pecuniary incentives are partially acknowledged by open source advocates. For example, Eric Raymond (2000b, Chapter 5) writes: "It can get you a better job offer, or a consulting contract, or a book deal". But, according to Raymond, this "is at best rare and marginal for most hackers."

[27] See Holmstrom (1999).

[28] Raymond (2000b). This is also emphasized in Point 4 of the Open Source Definition which says explicitly that "authors … have the right … to protect their reputations", http://www.opensource.org/docs/definition.html.

spread all over the world. It also highlights the individual contribution of each programmer. Finally, there is substantial evidence that contributing to open source did help many programmers to get access to venture capital or to be offered attractive jobs by commercial software developers.[29]

Finally, some programmers who devote their time to the development of open source software are employed by commercial software companies. These companies have an incentive to invest in open source development, if the open source software is a complement to the commercial software or hardware they produce. Some firms such as Red Hat, VA Linux or SuSE provide services and products that are complementary to Linux but that are not efficiently supplied by the open source movement. If Linux is improved and becomes more widely adopted, these firms get access to a larger market. Similarly, since server hardware and server operating systems are complements, server vendors who use operating systems developed by others benefit. Their profits increase if an open source operating system such as Linux which is free (in contrast to Microsoft's Windows) gains a larger market share.[30] However, these indirect benefits, that can be captured by commercial firms, are small as compared to the direct benefits of new or improved open source software that fully accrue to consumers. Therefore, the commercial companies do have a strong incentive to free ride on the contributions to open source by others, and their subsidies to OSS development are likely to remain limited.

Comparing the incentives to innovate for proprietary and open source software developers it has to be acknowledged that the open source movement does provide some incentives for innovations that proprietary software developers find it difficult to mimic. First of all, commercial software developers cannot make the source code available to everybody if they want to protect their intellectual property rights. Thus, with proprietary software it is impossible for users to fix bugs that they encounter or to customize the software to their own needs. Open source software lowers the cost of innovations by making the source code available and encourages large-scale parallel bug fixing.[31] Furthermore, as Lerner and Tirole (2002, p. 25) point out, “commercial companies will never be able to duplicate the visibility of performance

[29] For example, Lerner and Tirole (2002) report that the founders of Sun and Red Hat had signaled their talent in the open source world. Their Table 2 summarizes several other examples.
[30] There are several other more or less successful open source strategies employed by commercial firms. See Lerner and Tirole (2002) for a discussion.
[31] Some commercial software companies try to partially mimic open source by promoting widespread code sharing within the company and by making the source code available to some core customers (under restrictive conditions).

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 powerful incentives to existing employees." However, comparing the private benefits accruing to open source software 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 proprietary software developers to capture at least some of the fruits of their efforts, i.e. to turn at least some of the consumer surplus that is generated by their innovations into profit. Like 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 (1942, p. 110) 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 development of output; they were essentially one and the same thing or, as we may also put it, the former was the propelling force of the latter."

Twenty years later, Schmookler (1966, p. 199) summarized his empirical findings on what drives technological progress:

> "Despite the popularity of the idea that scientific discoveries and major inventions typically provide the stimulus for inventions, the historical record of important inventions in petroleum refining, paper making, 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 profitable opportunity to be seized; in short, a technical problem or opportunity evaluated in economic terms."

There is no reason to believe that the software industry differs fundamentally from all other industries when it comes to the incentives to innovate. If it had not been possible to

appropriate the profits of a risky project of software development, the rate of technological progress would have been much lower in the software industry over the past decades.

But it is not just the amount of effort and investments in innovation that is important. It is also the direction of technological change and how the innovations respond to what consumers want. Again, the open source mode has some advantages in this respect. In particular, it allows sophisticated users to develop and customize software to their individual needs.

However, there are also strong disadvantages of open source. A proprietary software developer has a strong incentive to respond to the needs of all potential users of her software. The more valuable the software is for her customers and the more consumers will benefit from it, the higher is the price that she can charge and the more copies of the software she will sell, so the higher is the profit that she is going to make. Therefore, a proprietary software developer has strong incentives to engage in market research in order to identify the needs of her customers and to adopt and develop the software as closely as possible to what consumers want.

This incentive is absent in the open source mode. Open source software developers are sophisticated users and IT professionals who respond to their own needs and the needs of other sophisticated users and IT professionals. Furthermore, they want to get recognition from their peers who are again sophisticated users and IT professionals. Writing software for unsophisticated end-users involves many tasks that may be intellectually unsatisfying (such as providing a user friendly interface or a detailed and easy-to-read documentation) and that are poorly suited to get peer recognition. Furthermore, the more applied a software is, the more important it is to have detailed knowledge about the background of the application. For example, in order to write a good accounting software, it is not only necessary to have good programmers, but detailed knowledge about the rules and requirements of accounting are at least equally important. This requires the close collaboration of IT professionals with professional accountants and other experts in several different fields. Similarly, the development of a sophisticated computer game requires the collaboration of programmers with graphic designers, marketing experts, etc. Even the development of an office software, such as a word-processor or a spread sheet, requires detailed research on the needs and preferences of the least sophisticated users. The open source mode does not provide sufficient incentives to engage in these 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 technicians and IT professionals, while it has been much less successful 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 proprietary mode and have been turned open source only after they failed commercially (e.g. the Star Office Suite or the Netscape Internet browser).

C. Network Effects and Switching Costs

It is often argued that there are market failures on the software market due to strong network effects and switching costs, and that these characteristics of the software market restrict competition. 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 open source software.

1. Network effects

Network effects arise if there is a complementarity between the adoption of a good by different customers: Additional adoption makes existing adopters better off (i.e., increases their *total utility* from adoption) and increases the incentive to adopt (i.e., increases their *marginal utility* from adoption). The software market is characterized by both "direct" and "indirect" network effects.[32]

- *Direct network effects* arise if my utility from using the good is directly increased if other people use this good, too. For example, if I want to share files with other people, it is important to me that my software and other people's software are compatible with each other.
- *Indirect network effects* arise if wider adoption benefits adopters by changing the behavior of third actors, e.g. sellers of that good, or sellers or buyers of some related goods. For

[32] 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.

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*:[33] If customer A decides whether or not to adopt a certain software, 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. each consumer wants to adopt A if all other consumers adopt A, even if this consumer would have preferred that everybody adopts software B.[34] 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 this 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 open source software is "forking". If software developers disagree on how a certain software should be developed, it sometimes happens that different groups of software developers develop different versions of a software that are not compatible with each other. A proprietary software developer who owns and controls his software can easily prevent this from happening by imposing which development path has to be followed. In contrast, open source software is not controlled by anybody, so it is much more

[33] 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 and Klemperer (2001, pp. 44-45).
[34] See Farrell and Klemperer (2001, p. 48).

difficult to prevent forking. The open source movement recognizes this problem and tries to solve it by imposing a "social norm" against forking.[35]

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 it is an equilibrium if all consumers buy software A, but it is also an equilibrium if all consumers buy software B. Which one is going to be adopted depends on which good consumers *expect* to be adopted. Even if everybody agrees that software A is strictly superior to software B, if everybody expects software B to be adopted, this expectation 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 which offers the highest surplus. This argument is particularly convincing if there is one large customer (or a large group of customers who can coordinate their behavior) who moves first.

- If consumers can coordinate their behavior (by writing contracts on which good to adopt or simply by cheap talk) 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.

But it is also possible that consumers coordinate on the wrong good, in particular if there is inertia in adoption. For example, it could happen that after good A has been widely adopted a

[35] Raymond (2000b) 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."

new good B is developed that offers a higher surplus on a clean-slate comparison. But if everybody expects everybody else to stick to good A, nobody is going to adopt the superior good B. However, if there is a large installed base of A and if consumer's 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 everybody sticks to A rather than switches to B.

If the network goods are differentiated and if some consumers prefer that everybody uses software A while some other consumers prefer that everybody uses 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 everybody expects 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 by everybody. 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 is going to win may be history dependent and may 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, this 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 the net effect is likely to be negative. Two networks can also coexist in the market if they are compatible with each other. In this case the conventional lesson applies that market power depends on the degree of product differentiation.

The discussion so far completely ignored pricing. There is a large literature on pricing with network effects, but this literature does not offer a clear conclusion.[36] 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 buy it repeatedly or if he will buy 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.[37] 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, in setting up a relationship with the seller, etc.[38] 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 a "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 an 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.

[36] See Farrell and Klemperer (2001) for a survey.
[37] See Farrell and Klemperer (2001, p. 12).
[38] 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.

However, the prospect of this rent will intensify competition for the first good and drive down its price below cost. This price pattern of "bargains followed by ripoffs" is familiar from many other goods with switching costs.[39]

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 is 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 profitable 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, and it seems unlikely that they can affect prices this way. Furthermore, it is inefficient because the relationship specific investments have to be duplicated.

The main concern about switching cost 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, again, the effect is ambiguous:

- 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 (the "fat cat" effect). Therefore, a new entrant with no customer base can enter the market and sell to new customers even if this entrant has higher costs than the incumbent. In this case there would be too much entry from a social 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 produce 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 existing customers.

[39] A well known example is the strategy to give away the razor for free in order to sell more razor blades.

If switching costs are large and real social costs, it is actually efficient that there is no market entry.

- Entry may also be too hard if switching costs are created artificially by the incumbent, for example if the switching costs are contractual (as with "loyalty" programs) or if the incompatibility 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 cost 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, that there is little actual switching taking place, and that 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 on switching-cost markets can be very 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 on 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 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": which software product is going to be 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 clear-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 anti-trust scrutiny is warranted if firms choose to make their products incompatible or to have 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 Germany an initiative called "Bundestux" of members of parliament from all major political parties declared that the "introduction of a free operation system in the Bundestag" to be a "necessary signal for Germany" for reasons of "basic regulation, competition and location policy, as well as for democratic reasons."[40] 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, one of whose mission is "to encourage administrations to use free software and open standards." The U.S. government has provided substantial support for R&D efforts that create open source software that must be released under the GPL.[41] Several Italian municipalities, including Florence and Pavia, have passed motions mandating the use of "software libero".[42] The government of Peru plans a bill that "makes it compulsory for all public bodies to use only free software".[43] The Taiwanese government has just passed a "National Open Source Plan" that requires all schools and public agencies to switch to open source software

[40] See heise online, "Tux Takes its Seat in Germany's Federal Parliament", published February 28, 2002, http://www.heise.de/english/newsticker/data/anw-28.02.02-006/ (accessed May 27, 2002).}
[41] See Thomas Sterling, "Beowulf Linux Clusters," http://beowulf.gsfc.nasa.gov/tron1.html (downloaded July 12, 2002) and Evans and Reddy (2002, Section VI.D) for several other examples.
[42] See Paul Festa (2001), Nations Uniting for Open Source, ZDNet, August 28, 2001, http://news.zdnet.co.uk/story/0,,t269-s2094089,00.html.
[43] See Networld, May 6, 2002, http://www.golem.de/0205/19653.html (accessed May 27, 2002).

within the next three years.[44] 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 open-source software to promote alternatives to current software and that it should encourage schools to take up Linux.[45] Finally, the European Commission has recently published an extensive study into the use open source software in the public sector.[46]

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. Some proponents of the open source movement, in particular those with a libertarian background, come to the same conclusion: They also deeply distrust any public intervention by bureaucrats and politicians and believe that the OSS "bazaar"[47] will succeed on its own merit. Public policy should be limited to promote compatibility standards that serve public needs, and not specific groups or corporate interests, including OSS.

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

A. Direct Subsidies for Specific Open Source Projects

The government could directly support a particular open source project either by offering direct subsidies to this project or by employing computer experts at universities or government agencies to support it. To be sure, in all developed countries a large fraction of R&D expenditures is paid for by the government. 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. It is typically 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

[44] Sueddeutsche Zeitung, June 5, 2002, p. 26.
[45] See http://www.statskonsult.no/publik/publikasjoner/2001-07/r2001-07eng.pdf for an English translation of the report.
[46] See http://www.idaprog.com/?http&&&ag.idaprog.org/Indis35Prod/doc/333 .
[47] This term is used by Eric Raymond (2000a), "The Cathedral and the Bazaar", who argues 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.

positive spillover effects of basic research are widespread and very difficult to internalize by commercial companies, so 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.

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 and it is much 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 is developed towards the needs of potential customers. A profit maximizing firm has a strong incentive to do so. The better its product is adapted to the needs of its customers, the more consumers are interested in buying the product and the higher is the price that can be charged for it. Thus, a commercial company will focus on what customers want, and it will try to do so in the most cost effective way. Only if it develops a better product at lower cost than its competitors, will it be able to survive on the market. In contrast, a government sponsored research lab does not face the constraints of the market and has much less incentives to focus on customer needs and cost efficiency.

Another problem with public subsidies to applied R&D is that it invites rent seeking activities. A project that wants to get public funds 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).

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 share

of the market. If consumers coordinate on the right product, this outcome is also efficient. However, which product is going to make it depends on the expectations of consumers that 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 other products out of the market. But governments have a poor track record in picking winners through "industrial policies". The government has neither the ability nor the right incentives to decide what the most efficient software product is. This 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 on software development, it should make sure 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 the BSD or other liberal licenses. 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 Open Source Software in the Public Sector

Several governments are currently considering prohibiting government agencies from using proprietary software if an alternative open source software exists and to force schools and universities to switch to open source.

The proponents of this policy argue that open source software 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 E-Mail-Server and Groupware solution. It argued that "Open Source solutions are yet insufficient for the requirements of the parliamentarians". 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 will largely be one based on Linux (whereas that for 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 open source software alone.[48]

Therefore, proponents of a move to open source sometimes rely on the additional argument that Open Source Software 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 on the market by favoring open source software.

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 share of the market. In this case, the government could make the market tip. In particular, if it forces schools and universities to adopt open source software, it will significantly lower the costs of students to learn how to work with OSS. If there are high switching costs to move to proprietary software afterwards, OSS may capture the entire market

[48] It is estimated to cost an additional Euro 80 000 per year. See heise online, February 27, 2002, „Open Source im Bundestag als `strategischer Vorteil'", http://www.heise.de/newsticker/data/odi-27.02.02-000/ (accessed June 2, 2002) and heise online, February 28, 2002, "Tux takes its Seat in Germany's Federal Parliament", http://www.heise.de/english/newsticker/data/anw-28.02.02-006/.

even if it is not qualitatively superior to proprietary software. Thus, it may be the government that picks the winner on this market. 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 open source software 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 fosters a development in which there are two incompatible networks on the market. This is inefficient, because significant positive network effects are lost, and it may reduce competition on 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 and to 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 one is open source software that is distributed for free. We can distinguish three groups of consumers on this market: Some consumers will always buy the proprietary software (e.g. because they are unsophisticated users who are unable to use the OSS product), some consumers will always go for the open source software (e.g. because they want to adopt the software to their specific needs) and 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 (who belong to the third group of consumers) to adopt the OSS product. This reduces the market size of the consumers for whom there is competition. Thus, the proprietary software developer has less of an incentive to reduce his price in order to compete for these customers, but will rather 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 anymore 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, on 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 quality. On the other hand, the incentives of the OSS developer are not directly affected. Her market size has increased, but she does not benefit financially from quality improvements anyway.

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, to increase prices, to reduce the incentives to innovate, and to make everybody worse off. In the Appendix, we present a little model that makes these arguments precise.

C. Subsidies for Institutions that Coordinate Open Source Development

The government 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, that is co-funded by the German federal government and private companies such as Hewlett-Packard and Linux Information Systems.[49] This policy may have some merit, in particular if the role of such an institution is neutral towards any particular open

[49] "The main goal of BerliOS is to support the different interest groups in the area of Open Source Software (OSS) and thereby to offer 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", see http://www.berlios.de/index.php.en , accessed May 31, 2002.

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 open source software.

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 the decision on which projects to promote will be biased by political pressures.

V. Conclusions

In this paper we have shown that open source software and proprietary software do have different impacts on the market outcome. Open source software 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 remain so. From a dynamic perspective, marginal cost pricing is inefficient, because it gives insufficient incentives to software developers to engage in R&D. To be sure, there are significant incentives for professional programmers to contribute to open source software, in particular to fix bugs and to adopt software to their particular needs. However, the social benefits of a new and innovative software product are far larger 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 open source software developers tend to be more responsive to the specific needs of other sophisticated users and IT professionals.

Therefore, despite the impressive record of open source software 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 has to make

its software as useful as possible to as many consumers as possible. As Adam Smith put it more than two centuries ago:[50]

> "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."

This powerful incentive is largely absent for open source software 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 has already a dominant market share by some measures (e.g. the Apache web server or Sendmail). For some other products it is still unclear what the superior solution is going to be. For these products, open source is putting strong competitive pressure on proprietary software developers. This is beneficial because it reduces prices and spurs competition in quality and innovation.[51]

However, the government should not interfere with the market and artificially favor any specific product or open source software as a whole. If there are strong network effects, the market equilibrium depends on consumers' expectations about future adoption decisions of other players. In this case a 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 this product may actually reduce competition, increase prices and lower innovation and social welfare.

Thus the government should restrict itself to subsidize basic research on software technology at universities and other academic institutions in order to promote new scientific developments that could be the basis for new products, no matter whether proprietary or open source, and to facilitate standard setting and encourage compatibility.

---

[50] Adam Smith (1776), Book 1, Chapter 2.

[51] Many empirical economists claim that there is an inverted U-shaped relationship between the degree of competition (as e.g. measured by the number of competitors) and the incentives to innovate (see Scherer and Ross, 1990, for a survey). Thus, incentives are strongest if there are just a few firms who fiercely compete for the market. See Schmidt (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 on which two software products are offered, proprietary software (PS) and open source software (OSS). The proprietary software producer maximizes profits, while the open source software 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 open source software, 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

$$U_u(PS) = v_p - t \cdot x - p \tag{1}$$

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

$$U_u(OSS) = v_o - t \cdot (1 - x) \tag{2}$$

where v_0 is her gross utility from the open source software 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.[52] Thus, the marginal consumer, $\bar{x}$, who is just indifferent between buying PS or OSS is characterized by

[52] This is the case if it is neither optimal for the proprietary software developer to focus just on customer group N_p, which requires that

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

and that it is not optimal for the proprietary software developer to capture all of the undecided customers, which requires that

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

which yields

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

Thus, the proprietary software developer maximizes

$$\Pi_p = p[N_p + \bar{x}N_u] \tag{5}$$

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

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

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

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

and

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

Suppose now that the government decides to force some agencies or schools and universities to adopt the open source software, 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 goes up. 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)

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

Both conditions are satisfied if, e.g., N_p is not too large as compared to N_u and v_p-v_o is not too large as compared to t.

with respect to N_u yields

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

and

$$\frac{d\bar{x}}{dN_u} = \frac{(v_p - v_o + t)4tN_u - 4t[N_u(v_p - v_o + t) - 2tN_p]}{8t^2 N_u^2} > 0 \tag{10}$$

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, she will raise prices and lose market share.

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

$$\frac{d\Pi}{dN_u} = \frac{2[(v_p - v_o + t)N_u + 2tN_p](v_p - v_o + t)8tN_u - 8t[N_u(v_p - v_o + t) + 2tN_p]^2}{[8tN_u]^2}. \tag{11}$$

This term is positive if and only if

$$N_u(v_p - v_o + t) > 2tN_p \tag{12}$$

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 suffer also from higher prices and less competition on the market for undecided customers.[53]

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

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

[53] 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.

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

$$\frac{d\Pi_p^*(y)}{dy} = \frac{N_u(v_p + y - v_o + t) + 2tN_p}{4t} - c'(y) = 0. \tag{14}$$

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

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

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

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

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 her 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 open source software. 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 open source software need not increase competition and foster the development of new and better software, but it may have the exact opposite effects. 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.

References

Aghion, Phillipe, and Peter Howitt, 1992, A Model of Growth through Creative Distruction, Econometrica, Vol. 60, 323-351.

Baumol, William J., Panzar, John C., and Willig, Robert D., 1982, Contestable Markets and the Theory of Industry Structure, New York: Harcourt Brace Jovanovitch.

Bessen, James, 2001, Open Source Software: Free Provision of Complex Public Goods, mimeo, Research on Innovation.

Coase, Ronald H., 1972, Durability and Monopoly, Journal of Law and Economics, Vol. 15, 143-149.

Demsetz, Harold, 1968, Why Regulate Utilities?, Journal of Law and Economics, Vol. 11, 55-65.

Evans, David S., 2001, Is Free Software the Wave of the Future? The Milken Institute Review.

Evans, David S. and Bernard Reddy, 2002, Government Preferences for Promoting Open-Source Software: A Solution in Search for a Problem, mimeo, NERA.

Farrell, Joseph and Klemperer, Paul, 2001, Coordination and Lock-in: Competition with Switching Costs and Network Effects, mimeo, Oxford University, http://www.paulklemperer.org/index.htm

Fehr, Ernst, and Klaus M. Schmidt, 2002, Theories of Fairness and Reciprocity - Evidence and Economic Applications, in: Dewatripont, M. et.al., Advances in Economic Theory, Eigth World Congress of the Econometric Society, Cambridge: Cambridge University Press

Grossman, Gene, and Elahan Helpman, 1991, Innovation and Growth in the Global Economy, Cambridge: MIT Press.

Gul, Faruk, Sonnenschein, Hugo, and Wilson, Robert, 1986, Foundations of Dynamic Monopoly and the Coase Conjecture, Journal of Economic Theory, Vol. 39, 155-90.

Harhoff, Dietmar, Henkel, Joachim, and Eric von Hippel, 2000, Profiting from Voluntary Information Spillovers: How Users Benefit by Freely Revealing their Innovations, mimeo, University of Munich.

Holmstrom, Bengt, 1999, Managerial Incentive Problems: A Dynamic Perspective, Review of Economic Studies, Vol. 66, 169-182.

Johnson, Justin P., 2001, Economics of Open Source, mimeo, Cornell University.

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

Mockus, Audris, Fielding, Roy T., and James Herbsleb, 2000, A Case Study of Open Source Software Developement: The Apache Server, http://www.research.avayalabs.com/user/audris/papers/apache.pdf , accessed May 31, 2002.

Raymond, Eric, 2000a, The Cathedral and the Bazaar, http://www.tuxedo.org/~esr/writings/homesteading/cathedral-bazaar/, accessed May 31, 2002.

Raymond, Eric, 2000b, Homesteading the Noosphere: An Introductory Contradiction, http://www.tuxedo.org/~esr/writings/homesteading/homesteading/, accessed May 31, 2002.

Romer, Paul M., 1990, Endogenous Technological Change, Journal of Political Economy, Vol. 98, 71-102}

Scherer, Frederic M. and David Ross, 1990, Industrial Market Structure and Economic Performance, 3rd edition, Boston: Houghton Mifflin Company.

Schmidt, Klaus M., 1997, Managerial Incentives and Product Market Competition, Review of Economic Studies, Vol. 64, 191-214.

Schmookler, Jacob, 1966, Invention and Economic Growth, Cambridge: Harvard University Press.

Schumpeter, Joseph, 1942, Capitalism, Socialism and Democracy, New York: Harper.

Smith, Adam, 1776, An Inquiry into the Nature and Causes of the Wealth of Nations, London: Strahan and Cadell, reprinted Newy York: Penguin Putnam, 1982.

Sobel, Joel, 1991, Durable Goods Monopoly with Entry of New Consumers, Econometrica, Vol. 59, 1455-85.

von Hippel, Eric, 1988, The Sources of Innovation, New York: Oxford University Press.

Willig, Robert D., 1976, Consumer's Surplus without Apology, American Economic Review, Vol. 66, 589-597.