GOVERNMENT PREFERENCES FOR PROMOTING OPEN-SOURCE SOFTWARE: A SOLUTION IN SEARCH OF A PROBLEM

David S. Evans and Bernard Reddy

NATIONAL ECONOMIC RESEARCH ASSOCIATES

ONE MAIN STREET
CAMBRIDGE, MASSACHUSETTS 02142
TELEPHONE: *617.621.0444* FACSIMILE: *617.621.0336*
INTERNET: *http://www.nera.com*

May 21, 2002

The authors would appreciate any comments or suggestions you might have on this working paper. Please e-mail to david.evans@nera.com and bernard.reddy@nera.com.

A version of this paper and abstract may be viewed at http://ssrn.com/abstract_id=313202.

n/e/r/a

Consulting Economists

Table of Contents

page

List of Figures

List of Tables

GOVERNMENT PREFERENCES FOR PROMOTING OPEN-SOURCE SOFTWARE: A SOLUTION IN SEARCH OF A PROBLEM

By

David S. Evans and Bernard Reddy*

I. Introduction

Governments around the world are making or considering efforts to promote open-source software (typically produced by cooperatives of individuals) at the expense of proprietary software (generally sold by for-profit software developers).[1] Proposals include having government agencies standardize on using open-source software, providing procurement preferences to open-source software, and subsidizing research and development of open-source software. The European Parliament, for example, adopted a resolution in September 2001 that calls on the Commission and Member States "to promote software projects whose source text is made public."[2] The German Bundestag is considering legislation that would require government agencies to use open source.[3] Former French Prime Minister Jospin created an agency whose mission will be to "encourage administrations to use open

* Evans is with NERA Economic Consulting in Cambridge, MA and the Center for the New Europe in Brussels, Belgium. Reddy is with NERA Economic Consulting in Cambridge, MA. We are grateful for financial support for our research from Microsoft. We also thank Robert Hahn and Anne Layne-Farrar for helpful comments and James Hunter, Bryan Martin-Keating, and Irina Danilkina for exceptional research assistance.

[1] We use the term "open-source" to refer to software that is made readily available in the form of source code (see Section II for a further discussion of concepts related to open source).

[2] The European Parliament Resolution on the Existence of a Global System for the Interception of Private and Commercial Communications, September 5, 2001, http://www3.europarl.eu.int/omk/omnsapir.so/pv2?PRG=CALEND&APP=PV2&LANGUE=EN&TPV=PROV&FILE=010905 (ECHELON link, downloaded May 20, 2002).

[3] "Digitale Spaltung der Gesellschaft Überwinden—Eine Informationsgesellschaft für Alle Schaffen," June 20, 2001, http://dip.bundestag.de/btd/14/063/1406374.pdf (downloaded March 30, 2002). This legislation differs from the Bundestag's decision in the March 2002 to use open source programs such as Linux for some of its own IT needs. See Section VI below.

source software and open standards."[4] The U.S. government has supported R&D efforts that create software that must be released under restrictive open-source licenses.[5] Leaders of the open source movement are naturally spurring these efforts.[6] But so are academics such as Professor Lawrence Lessig of Stanford Law School.[7]

This article examines the economic basis for these kinds of government interventions in the market. In the last twenty years, most governments have chosen to increase their reliance on market forces to govern the production and distribution of goods and services. Some previously communist countries, such as Poland, have sharply reduced centralized planning, privatized major national industries, and introduced market competition. Some capitalist countries, such as the United Kingdom, have reduced their reliance on government regulation and attempted to increase the scope of market competition. Policymakers are generally more skeptical than they were twenty years ago about the wisdom of having governments control markets—there remains a wide spectrum of beliefs, but the spectrum has shifted. Industrial policy—having governments pick winners and losers—has also lost some of its luster now that the success stories of the 1980s—Japan in particular—have become mired in seemingly interminable

[4] "Décret no 2001-737 portant Création de l'Agence pour les Technologies de l'Information et de la Communication dans l'Administration," August 23, 2001, http://www.legifrance.gouv.fr/citoyen/jorf_nor.ow?numjo=PRMX0105055D (downloaded April 3, 2002).

[5] Thomas Sterling, "Beowulf Linux Clusters," http://beowulf.gsfc.nasa.gov/tron1.html (downloaded April 5, 2002).

[6] See, for example, Free Software Foundation Press Release, "Richard Stallman Inaugurates Free Software Foundation-India, First Affiliate in Asia of the Free Software Foundation," July 20, 2001, http://www.gnu.org/press/2001-07-20-FSF-India.html (downloaded April 16, 2002); Free Software Foundation Press Release, "Richard M. Stallman Addresses Brazilian Congress on Free Software and the Ethics of Copyright and Patents," March 20, 2001, http://www.gnu.org/press/2001-03-20-Brazil.txt (downloaded April 16, 2002); and Cara Garretson, "Open Source Subject to Fiery Debate," *InfoWorld*, April 12, 2002, http://staging.infoworld.com/articles/hn/xml/02/04/12/020412hnopensource.xml?Template=/storypages/printfriendly.html (downloaded April 16, 2002).

[7] Lawrence Lessig, *The Future of Ideas*, Random House: New York, 2001, p. 247. See also Eben Moglen, "Anarchism Triumphant: Free Software and the Death of Copyright," *First Monday*, August 1999; Shawn W. Potter, "Opening Up to Open Source," *Richmond Journal of Law and Technology*, Spring 2000, p. 24;"Report to the President of the United States: Developing Open Source Software to Advance High End Computing," President's Information Technical Advisory Committee, October 2000.

recessions. Against this backdrop, it is a bit surprising to see so many countries entertaining policies to promote a particular method for producing and distributing technology.[8]

Although economists often have myriad opinions on the merits of any particular government intervention, there is a consensus about the principles that one should follow in determining whether an intervention is desirable.[9] First, economists insist on the identification of a significant market failure—a significant flaw or breakdown in the market process that prevents competition from giving consumers the greatest possible benefits given scarce resources.[10] Second, economists want some assurance that solving the market failure through government intervention will actually improve things (i.e., make consumers better off). That depends on whether it is possible to devise an intervention that solves the market failure without imposing direct and indirect costs that swamp the benefits.

This article examines whether there is evidence of a significant market failure in the production of software, and whether the proposals being considered might provide a cost-effective solution to such a market failure.[11] We reach the following conclusions:

- There is no evidence of a significant market failure that needs fixing.
- There is no reason to believe that the proposed government policies would actually increase social welfare.

---

[8] Other examples of market interventions with recent seemingly global appeal include efforts to promote "clean" technologies and efforts to reduce pharmaceutical prices (sometimes by voiding patent rights). Resolutions in the Netherlands and the United Kingdom provide subsidies to users of renewable energy sources, and the French and Russian governments have approved several decrees to regulate drug supplies to reduce prices. Paul Brown, "Hewitt's £20m will end solar power eclipse," *The Guardian*, March 26, 2002, http://www.guardian.co.uk/business/story/0,3604,674067,00.html (downloaded March 30, 2002); "Energy Market Trends in The Netherlands 2000," http://www.ecn.nl/unit_bs/emt/2000/part1.html (downloaded March 30, 2002); "Industry suffers as European governments target pharmaceutical spending," http://www.inpharm.com/intelligence/urch011001.html (downloaded March 30, 2002); and "Russian Government Takes Measures to Regulate Pharmaceutical Market," http://www.bisnis.doc.gov/bisnis/isa/9902phar.htm (downloaded March 30, 2002).)

[9] See, for example, Joseph E. Stiglitz, *Principles of Microeconomics*, New York: W. W. Norton and Company, Inc., 1997, p. 506.

[10] See *infra* notes 166 and 167.

[11] It does not deal with some of the ideological arguments for supporting open source—such as the Free Software Foundation's belief that proprietary software "pollute[s] our society's civic spirit."Richard Stallman, "Why Software Should Not Have Owners," 1994, http://www.gnu.org/philosophy/why-free.html (downloaded March 18, 2002).

The article is organized as follows. Section II provides background on software code and the role of intellectual property in protecting investments in code. This helps clarify issues concerning incentives that individuals and firms have for developing proprietary and open-source software. Section III describes the industrial organization and performance of the proprietary software business; the industry is performing well, and there is no obvious market failure that needs fixing. Section IV describes how the open-source movement produces and distributes open-source software. An understanding of these issues is needed to analyze some of the proposed government policies and how they would (or would not) affect the development of open-source software. Section V looks at advantages and disadvantages of the open-source and proprietary approaches to software and discusses the possible evolutionary paths for the software business in the absence of government intervention. Again, this is needed to analyze some of the proposed government policies and their possible effects on the development of open-source software. Section VI surveys government programs and proposals to promote open-source software and then analyzes these programs and proposals using the market-failure framework discussed above. Section VII presents brief conclusions.

II. Software Design and Intellectual Property Protection

Many things go into the creation of computer software before there is any code. The creation of a software package follows some conception of the purpose of the package—for example, to check spelling in documents, to determine whether a number is prime, or to control a particular piece of hardware. Going from the conception to a workable package requires the producers to develop architecture for the software that will guide programmers in the coding process. It may also require special numerical algorithms or programming tricks. As a result, a software package may depend on certain intellectual property, protected by patents or trade secrets that is independent of the code that is actually written. For example, the popular "MP3" audio format is based on patented algorithms held by Fraunhofer IIS-A.[12]

Most programs are written in one of several "high-level" languages whose commands look like a written language (usually English) and have meanings that are consistent with

[12] "Mp3licensing.com—About Us—Fraunhofer Gesellschaft," 2001, http://www.mp3licensing.com/about/fhg.html (downloaded March 19, 2002).

written language. For example, "If" and "While" are common commands in many languages. The commands in high-level languages provide a shorthand for more detailed instructions that are given to the computers—they enable the programmer to avoid many repetitive tasks. Popular high-level languages include C, C++, Java, Visual Basic, and Pascal. For example, Windows is written in C and C++, many custom applications written by corporate programmers for their companies' internal use are written in Visual Basic, most computer games are written in C or C++, much server-side "business logic" for Web sites is written in Java, and Linux (a popular open-source operating system) is written predominantly in C. These languages have "compilers" that translate the commands into binary code—a series of 1s and 0s—that the computer hardware understands.[13]

Figure 1 shows a simple example of a program. The first panel shows the design of the program. The second panel shows the C++ code that accomplishes the purposes of this design. The third panel shows the binary code that results from compiling this program.[14]

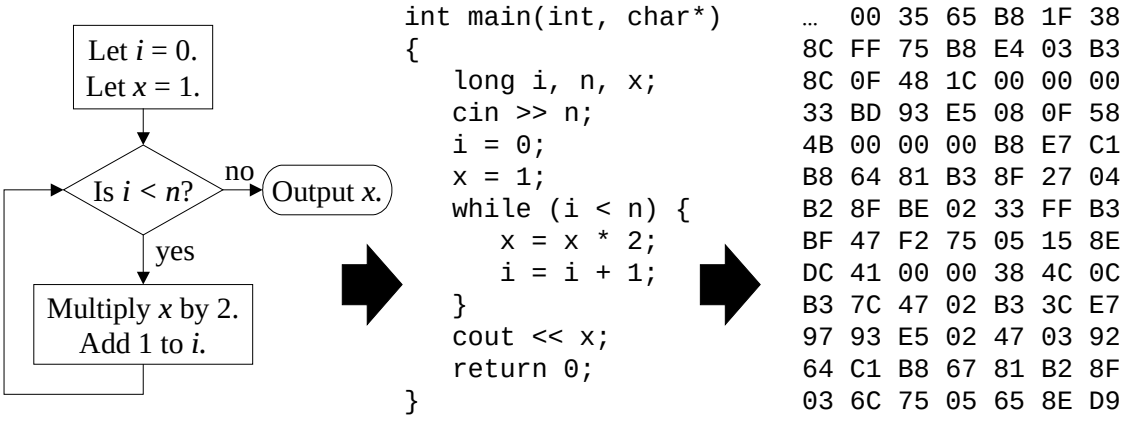


Figure 1. Simple Program: from Design, to Source Code, to Binary Code

The value of a program resides in the high-level software code (source code) that accomplishes the objective of the program (second panel) as well as the architecture, algorithms, and other elements that help the programmers write the code. In other words, if a programmer had the information in the left pane of Figure 1, he would have a significant head

[13] Some languages have "interpreters" rather than compilers, but the difference is not relevant for this article.

[14] The binary code is written in hexadecimal, which provides a compact way of writing sequences of 0s and 1s.

start in writing software code that accomplished the objective of the package. If a programmer had just the source code in the middle pane, as he would with open source programs, he would probably be able to figure out the architecture, algorithms, and other helpful tips for writing the code. He would also have the source code that he could compile and run. But if a programmer had only the binary code in the right pane, as he would with the typical proprietary program, he would have a very difficult time figuring out the source that generated that binary code or any of the intellectual property that was relied on in creating that binary code.[15]

The creators of software can prevent others from using their software—and any intellectual property embodied in that software—through various mechanisms. It is possible to obtain patents on algorithms and other creative aspects of software design. It is also possible to obtain a copyright on the software code (binary and source) that would prevent others from either copying the code without permission or using the code as an input into a product of their own. Finally, and perhaps most importantly, it is possible to limit seriously the ability of others to discover the intellectual property of software—including the source code—by distributing the software only in binary form. Creators of software can decide to rely on none, some, or all of these forms of intellectual property protection. Although there are examples along this full continuum of protection, copyright protection and trade secrets maintained by distributing binary code are the most common forms of protection.[16]

Two distinctions are useful for summarizing the major types of software. The first is the distinction between distributing software as source code ("open source") or as binary code

[15] Difficult but not impossible. Several attempts are underway to clone existing software by reverse engineering. For example, the WINE project is attempting to clone enough parts of Windows so that Windows applications can be run without change on Unix-like operating systems. ("WINE Development HQ—About," http://www.winehq.com/about.shtml (downloaded March 19, 2002).)

[16] Software vendors and developers typically do not "sell" their software to users. Rather, they retain copyrights (and other IP protection) and "license" the use of the software. This is common in IP-base industries. For example, someone purchasing a DVD of a hit movie has limited rights in the DVD: he cannot make copies for sale to others, nor can he display the movie in a commercial theater.

("proprietary").[17] The second is the distinction between distributing software at no direct charge ("unpaid"[18]) and for a direct charge ("paid").[19]

Over the years many people and groups of people have written software for which they have chosen not to exercise any property rights and have even made efforts to distribute the software widely.[20] Before the Internet made distribution easy, the source code for such software might be distributed on paper or electronic media such as computer tapes and disks. For example, programs to implement various mathematical functions were widely available in the scientific and technical communities. In other cases people have not distributed the source code but have freely distributed binary code and have not enforced any limitations on further distribution and use. This type of software is usually called "freeware."[21]

The spread of personal computers in homes and offices created a mass market for software. Not surprisingly, many commercial companies sprang up to write software for profit. These companies initially protected their investments by enforcing copyrights over their software and by distributing the software in binary form to prevent reverse engineering. By the mid 1980s, it became possible to obtain patents on certain aspects of software and many companies chose this route for protecting their intellectual property.[22] The number of software patents awarded annually to U.S. inventors has increased from 829 in 1986 to 7,398 in 2000.[23]

---

[17] Definitions of "open source" software are more complicated than is suggested by this simple dichotomy, which is sufficiently precise for current purposes.

[18] We avoid using the word "free" in this context because the term "free software" denotes a specific type of "open source" software.

[19] We say "direct" because sometimes software producers realize revenues indirectly from selling complementary products or services.

[20] This is somewhat analogous to what many academics do with their writings. Academics try to distribute their writings widely—often incurring costs to produce and send papers around—to get recognition for themselves and their ideas. Recognition is the currency of the realm and worth more than what academics could ever get from selling their writings. Obviously, if people were willing to pay (much) for academic writings, this pattern would change. And, indeed, scientific writings that contain valuable intellectual property are not distributed widely without proper intellectual property protection.

[21] The term "shareware" is typically applied to software distributed freely in binary form but for which the copyright holder requests that users pay for the product, possibly after an initial trial use period. In the 1980s, famous shareware programs included word processors (PC-WRITE), databases (PC-FILE), and communications programs (ProComm). Both freeware and shareware still exist today.

[22] According to Kortum and Lerner, "In software, there is no single clear-cut decision that opened the floodgates for patenting. Instead, we note that in [our] sample, the pace of patenting of [software] firms is trivial prior to

(continued...)

Most, but not all, proprietary software is sold at a price, and the source code is usually not made available. The mathematical package Mathematica is licensed to commercial users for $1,495 or $2,495, depending on the platform, and is available for many different kinds of computing platforms.[24] Intuit's QuickBooks accounting package costs between $180 and $500 depending on the edition.[25] Collections of "utility" programs for Windows (such as Norton Utilities) typically are priced in the range of $40-$70. *Final Fantasy X*, a popular game for Sony's PlayStation 2 console, sells for around $50 a copy.

What is now called "open-source" software is distributed under very different terms than is typical proprietary software. The source code is protected by copyright. However, it is distributed under a license that enables people to use the source code only if they comply with certain conditions. Perhaps the oldest open-source license, the BSD license,[26] has been modified over time, but it has typically allowed people to use the source code for free so long as they acknowledge the original copyright. People who modify the source code can choose to redistribute the binary code, the source code, both, or neither. For example, early versions of Sun's variant of the Unix operating system were based on a BSD version of Unix; the latest version of the Macintosh operating system is also based in part on a BSD version of Unix.[27]

(...continued)

1986" (Samuel Kortum and Josh Lerner, "Stronger Protection or Technological Revolution: What Is Behind the Recent Surge in Patenting?" *Carnegie-Rochester Conference Series on Public Policy,* Volume 48, June 1998, pp. 247-304).

[23] Count of patents awarded to U.S. inventors with International Patent Classification assignment to subclass G06F ("Electric Digital Data Processing "). Data purchased from Delphion, http://www.delphion.com/.

[24] "Wolfram Research—Shopping Cart," http://store.wolfram.com/checkout.cgi (downloaded March 19, 2002).

[25] "QuickBooks Basic 2002: Pricing," http://www.quickbooks.com/products/basic/pricing.html (downloaded March 19, 2002); "QuickBooks Premier 2002: Pricing," http://www.quickbooks.com/products/premier/pricing.html (downloaded March 19, 2002).

[26] This license was created by the University of California at Berkeley, primarily for distribution of Unix-related software, some of which its researchers had developed as a result of a contract with AT&T (the original owner of Unix).

[27] Peter H. Salus, *A Quarter Century of UNIX*, Reading, MA: Addison-Wesley, January 1995; "UNIX Based: The Open Desktop," http://www.apple.com/macosx/technologies/darwin.html (downloaded March 19, 2002).

Importantly, people can charge for the modified code and need not give away their modifications.[28]

Although many widely used open-source software products have been distributed under a BSD-style license, the General Public License (GPL) has become the dominant form of licensing for open-source software[29]—in part because the key persons and institutions that are spearheading the open-source movement believe that this is the license that will best achieve their goals (discussed below in Section IV.A.1).[30] If a program is distributed under the GPL, the source code for the program is made available. Others have the right to modify the program's source code for their own use without restriction and without charge. But anyone who distributes a modified version of a GPL program must likewise release the modified program under the GPL—and he must make the source code for the modified program available, essentially for free.[31] That precludes a firm from building a proprietary product based in part upon software licensed under the GPL—all enhancements to a GPL program must be made available to customers and competitors alike. Because of this feature, the GPL is sometimes called "viral"—programs that rely on software licensed under the GPL are "infected" by the GPL. Linux is perhaps the most popular software released under the GPL.[32]

---

[28] Another example of a company that does just that is Sendmail, Inc., which modifies the open-source "Sendmail" e-mail server software and sells its modified version. ("Sendmail—Company Overview," http://store.sendmail.com/cgi-bin/smistore/company.jsp?BV_UseBVCookie=Yes&filepath=overview/index.shtml&heading=Company%20Overview (downloaded March 30, 2002).)

[29] Hard information on the importance of various licenses is not available. As discussed below, information from a major hosting site for open source projects (SourceForge) suggests that roughly half of the projects hosted at the site rely on the GPL at least in part (license information is not provided for all projects, and some projects have multiple licenses); of those projects for which information is available on the type of license, roughly 70 percent rely on the GPL at least in part.

[30] The goals underlying the GPL are discussed below.

[31] Anyone who receives a copy of the modified code distributed under the GPL can redistribute it. Since it is not possible to restrict the number of copies that this recipient and subsequent recipients make, there is no effective restriction on supply. The competitive price therefore tends to zero for software distributed under the GPL.

[32] The "key persons and institutions" that favor the GPL also favor the use of the term "GNU/Linux" rather than simply Linux, to emphasize that the operating system commonly called "Linux" relies on much more than just the Linux "kernel." (Richard Stallman, "Why GNU/Linux?" 2000, http://www.fsf.org/gnu/why-gnu-linux.html (downloaded March 19, 2002).) In keeping with common usage, we use the term "Linux" for both the operating system and the kernel.

The remainder of this article mainly considers two major types of software that are at the center of the public-policy debates: proprietary software and open-source software distributed under the GPL.[33] The public policy debate centers around them for several reasons: most of the widely used software today is proprietary; the GPL currently appears to be the most popular license for open-source software; and given the restrictions of the GPL, proprietary software and GPL software (unlike BSD software) face a potentially uneasy coexistence.

III. The Economics of Commercial Software

An analysis of government intervention in an industry should begin with an examination of that industry: is there a market failure that can be addressed by intervention? To that end, we provide an overview of the commercial software industry and how it has changed over time.

Worldwide commercial[34] software is a $171 billion business based on revenue.[35] Revenues from U.S.-based commercial software companies amounted to over $90 billion in 2000.[36] That makes the U.S. commercial software industry larger than the motion picture industry ($74 billion in 2000) and around a fifth the size of the auto industry ($427 billion in 2000).[37] Generally, software is a relatively inexpensive but critical input into the production of

[33] One other type is software developed by people or businesses for their own use. Most large companies (and many smaller ones) have written software applications that accomplish particular functions—accounting is a common example. This software is proprietary, but the company typically chooses not to license it to others. Given that such internal software is designed around firm-specific issues, it would typically provide little or no benefit to other firms (except perhaps competitors as a form of market intelligence).

[34] In this section, we use the term "commercial software" as if it is synonymous with "proprietary software licensed to others." Some commercial firms do attempt to develop open-source software to license to others; they are probably a negligible component of the total commercial software industry.

[35] IDC Report #25569, "Worldwide Software Market Forecast Summary, 2001–2005," September 2001, Table 1, pp. 1-2.

[36] IDC Report #25569, "Worldwide Software Market Forecast Summary, 2001–2005," September 2001, Table 4.

[37] Sherlene K. S. Lum and Brian C. Moyer, "Gross Domestic Product by Industry for 1998-2000," *Survey of Current Business*, Bureau of Economic Analysis and the U.S. Department of Commerce, November 2001, Table 8, p. 30, http://www.bea.doc.gov/bea/ARTICLES/2001/11november/1101gdpxind.pdf (downloaded March 26, 2002).

computing services. This section describes the structure and performance of the commercial software sector.[38]

A. Overview of the Commercial Software Business

The size of the software industry has increased dramatically over the past few decades. From 1988 to 2000, revenues from worldwide proprietary software increased from $35 billion to $171 billion (measured in 2000 U.S. dollars)—an annual growth rate of over 14 percent.[39] See Table 1. In the United States, the number of software firms more than doubled between 1992 and 1999, and the number of employees more than tripled.[40] IDC, the leading vendor of data on the software industry, commented in its 2001 forecast of the market for software that, "it is likely that there are more than 10,000 companies competing in the packaged software market…"[41] In the United States alone, the Census Bureau identified almost 9,000 software firms with over 300,000 employees in 1999.[42]

[38] For more information regarding the software industry, see Kenneth G. Elzinga and David E. Mills, "PC Software," *The Antitrust Bulletin*, Fall 1999, pp. 739-786.

[39] IDC Report #8324, "1993 Worldwide Software Review and Forecast," December 1993, Table 2, pp. 10-16; IDC Report #25569, "Worldwide Software Market Forecast Summary, 2001–2005," September 2001, Tables 1, 16, pp. 1-2, 164-165.

[40] U.S. Census Bureau, "Statistics of U.S. Businesses," http://www.census.gov/csd/susb/susb2.htm (downloaded March 30, 2002). Figures based on the Bureau's classification of "Prepackaged Software" (NAICS 5112 for data collected in 1997 and later; SIC 7372, prior to 1997).

[41] IDC Report #25569, "Worldwide Software Market Forecast Summary, 2001–2005," September 2001, p. 15.

[42] U.S. Census Bureau, "Statistics of U.S. Businesses," http://www.census.gov/csd/susb/susb2.htm (downloaded March 30, 2002). Figures based on the Bureau's classification of "Prepackaged Software" (NAICS 5112 for data collected in 1997 and later; SIC 7372, prior to 1997).

Table 1. Packaged Software Revenues (in millions)

	All Packaged Software		PC Software Only	
	Worldwide	U.S./North America	Worldwide	U.S./North America
1983	NA	$13,486	NA	$2,455
1988	$34,674	$34,190	NA	NA
2000	$171,310	$90,613	$77,455	NA

Note: All numbers are in year 2000 dollars.
"U.S./North America" means the location of the software vendor, not the customer.
1983, 1988 "U.S./North America" numbers are U.S. only.
2000 "U.S./North America" numbers are for North America.
2000 "PC software only" number combines revenues for "32-bit Windows" and "other single user" operating environments. This numbers include revenues from software sold for PC-based servers.
Source: IDC Report #4046, "1989 Software Review and Forecast" April 1989, pp. 6-7; IDC Report #8324, "1993 Worldwide Software Review and Forecast," December 1993, Table 2, pp. 10-16; IDC Report #25569, "Worldwide Software Market Forecast Summary, 2001—2005," September 2001, Tables 1, 16, pp. 1-2, 164-165.

IDC divides software into three categories: "system infrastructure software," which includes operating systems; "application development and deployment software," which includes programming tools as well as spreadsheets; and "application software," which includes applications such as word processors. As shown in Table 2, applications software has the largest share of revenue (45 percent), followed by operating systems and other system infrastructure software (31 percent), followed by application development software (23 percent).

Table 2. Worldwide Packaged Software Revenues by Software Category, 2000

Category	Revenues (Millions)	Percent of Total
Application development and deployment	$40,244	23%
Applications	$77,280	45%
Packaged enterprise applications	$50,232	29%
Other applications	$27,048	16%
Systems infrastructure software	$53,786	31%
Total	$171,310	

Note: "Application development and deployment" includes programming tools and spreadsheets; "enterprise applications" includes vertical applications—accounting, human resources, electronic engineering, etc.; "other applications" includes applications software other than enterprise applications; "systems infrastructure software" includes operating systems.
Source: IDC Report #25569, "Worldwide Software Market Forecast Summary, 2001—2005," September 2001, Table 1, pp. 1-2, 15.

Compared with many other industries, the software industry is relatively unconcentrated. One conventional measure of industry concentration is the total share of sales accounted for by the four largest firms. In 2000, the four largest firms in the proprietary

software industry accounted for 26.7 percent of total revenues.[43] According to the latest Census data, nearly 47 percent of all manufacturing industries have a four-firm concentration ratio greater than that of the software industry.[44]

A second measure of industry concentration is the Herfindahl-Hirschman Index (HHI).[45] HHIs can range from zero (a large number of firms with infinitesimal market shares) to 10,000 (a monopoly with 100 percent of the market).[46] In 2000, the HHI for the software industry was 244,[47] a relatively low HHI when compared with industries such as automobiles (2,506) or breakfast cereals (2,446).[48]

There is also a great deal of turnover among the leading firms indicating that firms generally have not had entrenched positions in the software industry overall. Of the top ten companies in 1990, five did not make the list in 2000, either because they went out of business, they were acquired by another company, or their share of software revenues dropped over the

[43] IDC Report #25569, "Worldwide Software Market Forecast Summary, 2001–2005," September 2001, Table 2, pp. 27-29.

[44] U.S. Census Bureau, "1997 Economic Census, Concentration Ratios in Manufacturing," http://www.census.gov/prod/ec97/m31s-cr.pdf (downloaded April 1, 2002). Figures based upon value of shipments for industries categorized at the 4-digit SIC industry level.

[45] The HHI is equal to the sum of the squared values of each firm's share of the market. For example, a market that consists of four firms with market shares of 35 percent, 30 percent, 20 percent and 15 percent would have an HHI equal to 2,750 (35 x 35 + 30 x 30 + 20 x 20 + 15 x 15).

[46] The Antitrust Division of the U.S. Department of Justice and the Federal Trade Commission consider industries with HHIs of less than 1,000 to be competitive and those with HHIs of 1,800 or greater to be cause for significant competitive concern." (U.S. Department of Justice and Federal Trade Commission, 1992 Horizontal Merger Guidelines (revised April 8, 1997), Section 1.5 Concentration and Market Shares, available at http://www.usdoj.gov/atr/public/guidelines/horiz_book/hmg1.html (downloaded April 1, 2002).). As a practical matter, an HHI under 2,000 is seldom a serious concern for the U.S. antitrust enforcement agencies.

[47] IDC Report #25569, "Worldwide Software Market Forecast Summary, 2001–2005," September 2001, Table 2, pp. 27-29. Figures based on worldwide packaged software revenue. This report provides firm-level data only for the 100 largest software vendors, covering 61 percent of packaged software. Firms ranked 80th to 100th largest each hold one percent of the market. For purposes of this calculation, we assume that the remaining 39 percent of the market is composed of similar firms whose market shares also equal one percent (an assumption that produces the highest possible HHI estimate).

[48] U.S. Census Bureau, "1997 Economic Census, Concentration Ratios," http://www.census.gov/epcd/www/concentration.html (downloaded April 1, 2002). Figures based on the top 50 firms in 1997 and the Bureau's classification of "Motor Vehicles" (NAICS 3361) and "Breakfast Cereals" (NAICS 31123). In fact, 55 percent of industries, calculated at the four-digit level, had a higher HHI than the software industry.

decade.[49] Compare this to the turnover in pharmaceuticals, another industry based on intellectual property. Eight of the ten leading pharmaceutical companies in 1990 were still in the top ten in 2000.[50]

B. Production Method

The production of commercial software consists primarily of initial costs. Software development is generally an iterative process, with the development of typical commercial software including the following steps:

1. Identifying customer needs. After starting with a good idea, the developer needs to learn which features customers are likely to value, which features are likely to be considered essential, how different user interfaces can make the software easier to use, and so forth. The importance of these aspects of software design may well differ substantially across different types of software; computer games and email server software are likely to be used by different customers, with different capabilities.

2. Designing the software. This generally includes high-level concepts, such as what major modules will do, how the modules will communicate with each other (and with other computers, if relevant), and so forth.

3. Coding, building, and testing. Programmers typically test their code frequently, often in small pieces. Large software systems go through frequent "builds," in which all the different modules that have been initially tested by the coders are collected together, "built" into the complete product, and then subjected to a battery of tests. The testing reveals flaws, which require recoding and sometimes redesign. Flaws can include "bugs" (errors that cause the program to behave in undesirable ways in some circumstances) and performance problems. Redesign sometimes involves the dropping or simplification of problematic features.

Once the design/code/build/test process is completed, the product is ready to ship (documentation is typically developed along the way). In general, the variable costs for each unit of software shipped are low: the cost of a CD and a slender manual will seldom exceed a

[49] IDC Report #8324, "1993 Worldwide Software Review and Forecast," December 1993, Table 2, pp. 10-16. IDC Report #25569, "Worldwide Software Market Forecast Summary, 2001–2005," September 2001, Table 2, pp. 27-29.

[50] IMS Health, "M&A Drives Decade of Change," April 25, 2001, http://www.ims-global.com/insight/news_story/0104/news_story_010425.htm (downloaded April 1, 2002) (citing *World Review*, 1990-2000).

few dollars.[51] Depending on the type of software and target customer, sales, marketing, and promotional expenses can take different forms. Software for large servers is often expensive (in the tens of thousands of dollars, or more), and is often sold through a direct sales force. Mass-market software for end users of PCs is often priced at less than $100 and might be sold through retail stores, over the Web, through computer manufacturers, via direct mail solicitation, and so forth.

Support costs can also vary widely, depending on the type of software. Complicated software (such as for large servers) might have a separate support agreement. Mass-market software for end users might provide for limited support via phone or email. Developers of mass-market software have incentives to design software in ways that will provide desired functionality without requiring support; one or two technical support calls can wipe out much or all of the margin on a product retailing for less than $100.

After a product ships, two processes often begin: maintenance work on the just-shipped product, primarily to fix bugs but sometimes to add new features; and the design stage for the next version of the software.[52] Games software might not, strictly speaking, have "new versions," but successful games typically have sequels.

One feature of the software industry emerges clearly from this review: the average costs of any given software product fall with volume. In general, a large fraction of total costs for any given product come from the design, coding, and testing stages. For PC software, the largest purely variable cost[53] will sometimes be support, sometimes materials and packaging. But for typical PC software, these costs are low compared with the usual development costs.

[51] The fat manuals of the past have largely been replaced by electronic documentation, available on CD or from the Web.

[52] It is not unusual for a software developer to have multiple versions of a product under simultaneous development, with feedback across the versions.

[53] Marketing and selling costs may be substantial, but they do not generally increase automatically with a spurt in sales (except for sales commissions and the like).

C. Commercial Business Model and Nature of Competition

As discussed above, the development of commercial software generally involves high initial costs and relatively low marginal costs. In order to stay in business, a successful firm must charge substantially more than marginal cost in order to cover its fixed costs.[54] Most software projects are losers in the marketplace, but the financial bonanza available for a winner gives firms incentives to invest.[55] Me-too products offering little functionality beyond what is in competing products are seldom attractive for commercial firms to develop, unless the development costs are exceptionally low. With a me-too product, price competition from other vendors can quickly destroy profitability. As a result, commercial software firms (like many firms in other industries) prefer to differentiate their products from those of their competitors.

Given the high first costs and low marginal costs of software, competition within any particular product category has at least some elements of a "natural monopoly": higher volume means lower average costs, which means profitability can be achieved at a lower price. This potential for "natural monopoly" is increased if a software category exhibits "network effects." Network effects can arise purely on the demand side: if most business users of computers use the same word processing program, then it makes it relatively easy to trade files, to transfer knowledge of how to use software from a job in one firm to a job in another, and so forth. Network effects can also arise on the supply side, such as through the availability of complements. The more applications that exist for a given software platform, the more desirable the platform is for users; the more users a software platform has, the more desirable the platform is for developers. Not all software categories exhibit strong network effects, but those that do provide the potential for particularly large rewards for a winning program—because these network effects provide substantial benefits to users. But category leaders can and do get replaced.[56] In a software category with strong network effects, competition typically

[54] If a firm produces complementary products, this need not be true for every product. For example, AOL gives away its access software and attempts to make money by selling its Internet service, by selling advertising, and by making financial arrangements with vendors that sell their own goods and services through AOL.

[55] Josh Lerner, "The Returns to Investments in Innovative Activities: An Overview and an Analysis of the Software Industry," in *Microsoft, Antitrust and the New Economy: Selected Essays*, ed. David S. Evans, New York: Kluwer Academic Publishing, 2002, p. 463.

[56] *See* David S. Evans, Albert Nichols, and Bernard Reddy, "The Rise and Fall of Leaders in Personal Computer Software," in *Microsoft, Antitrust and the New Economy: Selected Essays*, ed. David S. Evans, New York:

(continued...)

takes the form of dynamic competition *for* the market, rather than static price competition *within* the market.[57] As a result, the existence of a "dominant" firm in such a software category does not imply that some type of "market failure" exists that government intervention can (and should) try to fix. By no means do all software categories have this characteristic—lack of network effects and the existence of heterogeneous groups of customers can enable multiple software firms to coexist in the same category.

If a competitor could readily copy (or enhance) the features of a successful commercial software firm's products, the firm would face the possibility of short-run price competition and long-run loss of leadership. It would have no way to recover the fixed cost of investments in its products or to be rewarded for the risks it has borne in financing software creation and development. In order to prevent this, commercial firms typically use the types of protection for their intellectual property discussed above: copyrights, patents, and trade secrets. Licensing only binary code makes reverse engineering more difficult for competitors. Keeping close control of source code makes copying of features more difficult. The commercial software industry that has thrived over the past 30 or more years would not exist as we know it today without these types of protection for intellectual property.

1. Performance of Commercial Software

The commercial software industry has exploded since its early years. Quality-adjusted prices have fallen markedly in nominal terms, even before adjusting for inflation. Quality-adjusted output has soared. By any measure, the commercial software industry is succeeding in providing ever-more powerful products that customers are willing to pay to acquire. Hardware has improved enormously as well, of course. But the hardware improvements have needed complementary software improvements to provide their current range of benefits to consumers. In general, the explosive growth of the commercial software industry casts serious doubt on

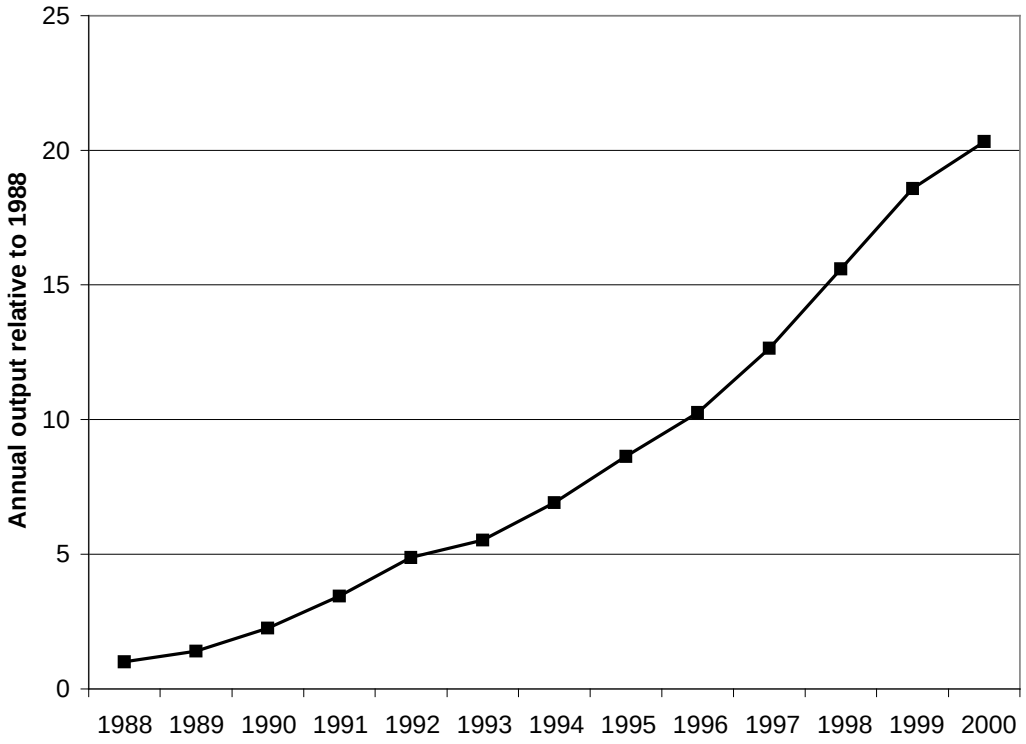
---

(...continued)

Kluwer Academic Publishing, 2002, p. 265; Stan J. Liebowitz and Stephen E. Margolis, *Winners, Losers & Microsoft: Competition and Antitrust in High Technology (revised edition),* Oakland, CA: The Independent Institute, 2001.

[57] *See* David S. Evans and Richard Schmalensee, "Some Economic Aspects of Antitrust Analysis in Dynamically Competitive Industries," in *Innovation Policy and the Economy, Volume 2* , edited by Adam B. Jaffe, Josh Lerner and Scott Stern, Cambridge, MA: The MIT Press, 2002.

any claims that a significant market failure in the industry needs to be cured by government intervention.



Source: IDC reports; "2001 Annual NIPA Revision," Bureau of Economic Analysis, and the U.S. Department of Commerce, August 2001, Tables 1, 11, http://www.bea.doc.gov/bea/papers/tables.pdf (downloaded March 21, 2002); and Robert Parker and Bruce Grimm, "Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts, 1959-98," *Survey of Current Business*, Bureau of Economic Analysis and the U.S. Department of Commerce, May 2000, http://www.bea.doc.gov/bea/papers/software.pdf (downloaded March 21, 2002).

Figure 2. Worldwide Quality-Adjusted Packaged Software Sales, 1988-2000

a. Output Has Boomed

Measuring "output" of the software industry is difficult, since the quality of software has improved enormously over time. The U.S. Bureau of Economic Analysis has constructed a price index for prepackaged software that attempts to control for quality improvements.[58]

[58] Robert Parker and Bruce Grimm, "Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts, 1959-98," *Survey of Current Business*, Bureau of Economic Analysis and the U.S. Department of Commerce, May 2000, http://www.bea.doc.gov/bea/papers/software.pdf
(continued...)

Based on this price index and IDC's estimates of annual packaged software revenues, Figure 2 shows the annual quality-adjusted output of the packaged software industry as an index relative to the level in 1988. In 2000, quality-adjusted worldwide output was more than 20 times as large as it was 12 years earlier. Similarly, quality-adjusted output by U.S.-based/North American software vendors increased 100-fold from 1983 to 2000 (data not shown).

b. Price and Performance Have Improved Markedly

The Bureau of Labor Statistics publishes a Consumer Price Index for "computer software and accessories," which is available only from December 1997 onward.[59] This index attempts to control for changes in software quality. From December 1997 through December 2001, the software CPI fell by 20.5 percent while the CPI for all items rose by 9.5 percent.[60] This means that the real price of software fell by approximately 27.4 percent over this period.

Computer hardware and software have both improved dramatically in terms of price relative to performance. One popular industry benchmark for database servers shows that the fastest system in April 2001 could process 60 times as many transactions per second as the fastest system in December 1996.[61] Based on system price relative to performance over that

---

(...continued)

(downloaded March 21, 2002). The BEA constructed this price index for "prepackaged" software by splicing together (over different time periods) annual percentage changes in a price index for computer hardware, a "hedonic" price index for spreadsheets and word processors, two different "matched model" price indices for different mixes of prepackaged software, and a producer price index for applications software (Table 6). The BEA also publishes price series for "custom" software and for "own-account" software. These two series are either flat or rise steadily, unlike the dramatic decline in the series for prepackaged software. This appears to be an artifact of the methodologies used by the BEA in constructing these other price indices. For example, the BEA assumes that there has been no quality improvement (and no productivity improvement) in the production of "own-account" software—the price index for that software is assumed to depend on compensation and office costs for programmers (Table 8; pp. 16-17). And annual changes in the price index for custom software are simply a weighted average of the annual changes in the price indices for prepackaged and own-account software (p. 17), with changes in the own-account software price index receiving a 75 percent weight.

[59] "Consumer Price Index—All Urban Consumers: Computer Software and Accessories," Bureau of Labor Statistics and the U.S. Census Bureau, March 20, 2002, http://data.bls.gov/servlet/SurveyOutputServlet?jrunsessionid=101666141318297318 (downloaded March 20, 2002).

[60] "Consumer Price Index—All Urban Consumers: All Items," Bureau of Labor Statistics and the U.S. Census Bureau, 1997-2001, http://data.bls.gov/servlet/SurveyOutputServlet (downloaded March 21, 2002).

[61] The Transaction Processing Performance Council (TPC) is a non-profit group that defines database benchmarks; its members include Compaq, HP, IBM, Intel, Microsoft, Oracle, SGI, and Sun. It administers a widely used benchmark, TPC-C, the results of which are publicly available; the benchmark measures both performance and

(continued...)

same time period, the most efficient system available improved by a factor of 12. Much of this improvement in performance (and price relative to performance) has been due to faster hardware. But if the 2001 hardware had been combined with the 1996 software, the 2001 performance would have fallen far short of the performance achieved with modern software.

c. Patents and Research and Development Have Expanded Sharply

Inputs to and outputs from the innovative process of commercial software development have also increased dramatically over the past 15 years. For example, in 1985, R&D expenditures by publicly traded software companies in the United States accounted for about 1 percent of total industrial R&D in the United States. By 2000, that number increased to 10 percent.[62] As mentioned above, patenting of software has also increased substantially.[63] In 1986, 829 patents were granted for software with at least one U.S. inventor. In 2000, 7,398 patents were granted.[64] Precise links between either R&D or patents and innovation may be difficult to trace, but the innovative process appears to be flourishing.

2. Lessons from the Vertical Disintegration of the Computer Industry

The proprietary software business owes much of its growth to fundamental changes in the extent to which hardware and software were integrated.

(...continued)

price/performance. TPC web site, http://www.tpc.org/tpcc/results/tpcc_results.asp?print=true&OrderBy=&version=3 (downloaded April 2, 2002).

[62] Data from Compustat. Software firms are those that report their primary SIC code to be 7372 (Software publishers).

[63] Some commentators have argued that it should not be possible to obtain patents on software. See generally Lawrence Lessig, *The Future of Ideas*, Random House: New York, 2001, pp. 205-217. Lessig himself argues against patenting software. He also cites others who have expressed skepticism about or outright scorn for software patents, including Richard Stallman and representatives from Adobe, Microsoft, and Oracle. Even Bill Gates has expressed skepticism about software patents, saying in 1991, "If people had understood how patents would be granted when most of today's ideas were invented and had taken out patents, the industry would be at a complete standstill today." Cited in Lawrence Lessig, *The Future of Ideas*, Random House: New York, 2001, p. 206.

[64] Data purchased from Delphion, http://www.delphion.com/. See note 23.

a. Software Production in the Mainframe Era

The 1960s and 1970s were the era of expensive mainframe and minicomputers.[65] The suppliers of these computers tended to be vertically integrated: they shipped their computers with operating systems, compilers for programming languages, and other software tools. In the 1960s, applications software was essentially customized. Firms that bought the large computers of that era generally either wrote their own software or hired outsiders to write software for them. By the late 1960s and early 1970s, commercial software had started to appear. In some cases, this consisted of software that a custom programming firm had written for one client, retained the rights to, and then licensed to other customers.[66] These products were typically expensive, although much less expensive (but possibly less versatile) than custom software.[67] By the end of the 1970s, off-the-shelf software for mainframes was commonplace. For example, SPSS and SAS, programs for data handling and statistical analysis, have been familiar to empirically minded social scientists since the mid-1970s.[68]

In the early parts of this era, when little or no commercial software was available, substantial trading of software was fairly common; the software was usually traded in the form of source code to make it easier to port from one computer to another. By the 1970s, according to Richard Stallman, substantial trading of software was unusual.[69]

---

[65] Several books discuss the state of computer hardware and software in the years just before the emergence of the personal computer: Paul Freiberger and Michael Swaine, *Fire in the Valley*, 2000; Steven Levy, *Hackers*, 1994; and Stephen Manes and Paul Andrews, *Gates*, 1994. Also, see the Web site for the Software History Center, http://www.softwarehistory.org/ (downloaded March 30, 2002).

[66] In the PC era, MultiMate had a similar origin; its original developer wrote a look-alike of the Wang word processor for an insurance company, then marketed the software generally. (Ken Polsson, "Chronology of Events in the History of Microcomputers," http://www.islandnet.com/~kpolsson/comphist.htm (downloaded September 1, 1998).)

[67] One veteran of that era has described starting a firm in 1971 to sell a payroll system, making 20-30 sales per year. (Luanne Johnson, "From Not-Invented-Here to Off-The-Shelf," 1997, http://www.softwarehistory.org/history/Johnson2.html (downloaded March 30, 2002).)

[68] "Our Company," http://www.sas.com/corporate/index.html (downloaded March 30, 2002); and "SPSS Inc. Corporate History," http://www.spss.com/corpinfo/history.htm (downloaded March 30, 2002).

[69] Richard Stallman, "Free Software: Freedom and Cooperation," speech delivered at New York University, May 29, 2001, http://www.fsf.org/events/rms-nyu-2001-transcript.txt (downloaded April 2, 2002).

b. Software Production in the PC Era

Mass-market software really dates from the emergence of small, relatively inexpensive personal computers in the late 1970s and early 1980s. Someone who wrote a useful program could now hope to sell copies to a large number of less sophisticated computer users. As a result, this era saw an explosion of proprietary software. In 1974, Gary Kildall created what was probably the first commercial operating system for personal computers, CP/M, and licensed it to many computer manufacturers.[70] Bill Gates and Paul Allen wrote their first version of MS-BASIC and formed Microsoft in 1975.[71] Word processing programs (such as Electric Pencil and later WordStar)[72] were among the first applications programs offered for sale, coming on the market in the late 1970s. The spreadsheet category of programs was invented with VisiCalc in 1979.[73] Database programs such as dBase II came along a few years later.[74] The open-source approach played no visible role in the early history of PC software. While early users of personal computers did swap software—Bill Gates raised the ire of hackers by complaining about their pirating of MS-BASIC[75]—there was no organized open-source movement.

The explosion of the personal computer hardware and software industries was accompanied by a major shift from the vertically integrated approach of mainframe and minicomputer vendors in earlier years. Some operating systems were still proprietary to hardware vendors (e.g., Apple has always shipped proprietary operating systems for its Apple II and Macintosh families of computers). The most successful operating systems, however—first CP/M, then MS-DOS, and later Windows—were licensed by software firms to

---

[70] Ken Polsson, "Chronology of Events in the History of Microcomputers," http://www.islandnet.com/~kpolsson/comphist.htm (downloaded September 1, 1998) (citing *BYTE*, June 1981, p. 224).

[71] The first version of MS-BASIC was for the MITS Altair. Paul Freiberger and Michael Swaine, *Fire in the Valley*, 2000, pp. 53-54; "Microsoft Timeline," http://www.microsoft.com/mscorp/museum/musTimeline.asp (downloaded March 30, 2002).

[72] Paul Freiberger and Michael Swaine, *Fire in the Valley*, 2000, pp. 187, 194.

[73] Paul Freiberger and Michael Swaine, *Fire in the Valley*, 2000, pp. 289-291.

[74] Ken Polsson, "Chronology of Events in the History of Microcomputers," http://www.islandnet.com/~kpolsson/comphist.htm (downloaded September 1, 1998).

[75] Paul Freiberger and Michael Swaine, *Fire in the Valley*, 2000, p. 195.

hardware vendors. The most popular development tools (such as language compilers) came from software firms, not hardware firms. And, of course, applications programs for personal computers have typically not been written by integrated hardware firms.

3. Summary

The commercial software industry has exploded over the last 20 years, providing a dizzying array of ever-more powerful software for use by both technically minded users and less sophisticated users. Quality-adjusted prices have fallen sharply. Overall firm concentration in the industry is low, and the identities of the top firms change over time. Leaders in major software categories frequently change as well. R&D and patent activity have risen substantially. There is no evidence that a significant market failure exists that is amenable to fixing by government policies toward procurement or R&D.

IV. The Economics of GPL Open-Source Software

An analysis of whether governments should intervene in the marketplace to support open-source software requires an understanding of what open source is about and of the incentives that various players may (or may not) already have to support the development of open source.

Software developed and licensed under the GPL shares many features with other types of open-source software, but it differs in some regards. For example, some observers have claimed that programmers are more inclined to donate their time to develop GPL software than other open-source software. And commercial firms may have different incentives for supporting the development of GPL software than for other open-source software with fewer restrictions on commercialization.

A. Institutional Arrangements

1. Free Software Foundation Is the Ideological Heart of the Movement

The current open-source movement can trace its origins to the Free Software Foundation (FSF), founded by Richard Stallman in 1985. The FSF initiated a project—known as GNU[76]—to develop a non-proprietary Unix-like operating system. The FSF's efforts were based in part on the belief that "free software is a matter of liberty."[77] According to Stallman, "Even if GNU had no technical advantage over Unix, it would have a social advantage, allowing users to cooperate, and an ethical advantage, respecting the user's freedom."[78]

Work began in the 1980s, by Stallman and the FSF, on the building blocks for an operating system. This effort started with development tools such as editors (Emacs) and compilers (GCC). In 1989, the FSF came out with the first version of the GPL. The GPL was designed to drive the software industry toward the "free software" model. As one FSF document points out:

> If we amass a collection of powerful GPL-covered libraries that have no parallel available to proprietary software, they will provide a range of useful modules to serve as building blocks in new free programs. This will be a significant advantage for further free software development, and some projects will decide to make software free in order to use these libraries. University projects can easily be influenced; nowadays, as companies begin to consider making software free, even some commercial projects can be influenced in this way.[79]

The GPL helps advance the FSF's goals by forming a kind of club. As the FSF views it, the people who develop software under its licenses are members of a special club; anyone wanting to distribute modified versions of the club's software must make the source code for the modified software available (essentially without charge) to the other members of the club:

---

[76] GNU stands for "GNU's Not Unix."

[77] Free Software Foundation, "The Free Software Definition," http://www.fsf.org/philosophy/free-sw.html (downloaded April 3, 2002).

[78] Richard Stallman, "The GNU Project," Free Software Foundation, http://www.fsf.org/gnu/thegnuproject.html (downloaded April 2, 2002).

[79] Free Software Foundation, "Why You Shouldn't Use the Library GPL for Your Next Library," http://www.fsf.org/philosophy/why-not-lgpl.html (downloaded April 2, 2002).

> We encourage two-way cooperation by rejecting parasites: whoever wishes to copy parts of our software into his program must let us use parts of that program in our programs. Nobody is forced to join our club, but those who wish to participate must offer us the same cooperation they receive from us.[80]

2. Copyleft and the Viral Aspect of the GPL Promote the Goals of the FSF

The GPL has been extremely important in shaping the development and evolution of open-source software. We now turn to a more detailed interpretation of the obligations the GPL imposes on people who use software covered by the GPL.

The viral nature of the GPL results from the following requirement, which is sometimes termed "copyleft":[81]

> 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.[82]

The copyleft provision means that if people choose to distribute software that is based in part on other software covered by the GPL, they must distribute their new software under the GPL. GPL software thereby propagates itself. The GPL is designed to prevent precisely what the BSD-style licenses expressly allow: modifying a program and distributing it without making public the modified source code and without granting others free use of the modified source code. Copyleft make it is essentially impossible for anyone to develop proprietary commercial software using code subject to the GPL.[83]

---

[80] Free Software Foundation, "The GNU GPL and the American Way," http://www.gnu.org/philosophy/gpl-american-way.html (downloaded April 2, 2002).

[81] Free Software Foundation, "What Is Copyleft?" http://www.fsf.org/copyleft/copyleft.html (downloaded April 2, 2002).

[82] The GPL can, it should be noted, affect patent rights as well as copyrights. If a firm (or programmer) has patent rights to the modifications it makes to a GPL program, then those rights (at least as embodied in the code in question) must be licensed without charge to others. Free Software Foundation, "GNU General Public License," http://www.fsf.org/copyleft/gpl.html (downloaded April 2, 2002).

[83] At least, not without explicit permission from the copyright holder. The owner of the copyright for a given program might choose to release the program generally under the GPL. At the same time, the owner could also license the program to another party on terms other than the GPL, such as normal commercial terms. This type of dual-licensing does occur, but the FSF warns the public that it rarely permits dual licensing of software whose copyright it owns. (Free Software Foundation, "Frequently Asked Questions about the GNU GPL," http://www.fsf.org/copyleft/gpl-faq.html (downloaded April 2, 2002). See especially "Using a certain GNU program under the GPL does not fit our project to make proprietary software. Will you make an exception for

(continued...)

Importantly, there is some ambiguity over when "contact" with a GPL program infects other programs. That is because the phrases "work … derived from the Program" and "based on the Program" (another term used in the license) are not clear. The license states explicitly that "mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License." But it also states that "derivative works" include software "containing … a portion of" a program covered by the GPL.

One interpretation is that using a single line of code from a GPL program in a new program is enough to qualify the latter as a "derivative work," requiring that it be licensed under the GPL.[84] Proprietary programs can use or communicate with GPL programs in some limited ways without themselves becoming subject to the viral license condition, but the FSF recognizes that the dividing line can be murky.[85]

The terms of the GPL apply only to the distribution of software licensed under the GPL, although what "distribution" means in this context is not entirely clear either. It may be possible for an enterprise to modify a GPL program and use it internally without being legally bound to make the source code for its modified version available to others. On the other hand, if the same enterprise distributed its modified GPL program to a subsidiary, the terms of the GPL might well require it to make the source code available to all comers.[86]

(...continued)

us? It would mean more users of that program.") Dual licensing becomes difficult to impossible to arrange if a GPL program has gone through many generations and has multiple copyright owners.

[84] This is true even if the GPL code is in a separate module that is "linked" (a technical computer term) to, rather than included directly in, another program.

[85] Free Software Foundation, "Frequently Asked Questions about the GNU GPL," http://www.fsf.org/copyleft/gpl-faq.html (downloaded April 2, 2002). See, especially, "What is the difference between 'mere aggregation' and 'combining two modules into one program'?" and "If a program released under the GPL uses plug-ins, what are the requirements for the licenses of a plug-in[?]."

[86] Ira V. Heffan, "Copyleft: Licensing Collaborative Works in the Digital Age," Stanford Law Review, Vol. 49, 1997, p. 1487; David McGowan, "Legal Implications of Open-Source Software," Social Science Research Network, December 2000, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=243237 (downloaded May 20, 2002).

B. Production of Open-Source Software

With some exceptions, open-source software has primarily been developed by individuals who donate their time to work on projects that interest them. In a typical situation, someone (or a small group of people) gets an idea for a project that is interesting, useful, or both. The original developers begin work on the project and eventually solicit support from other interested programmers. Over the course of the project, programmers, including the original developers, may come and go as they complete work and as their interests wax or wane. The programmers communicate with each other over the Internet. A core group, often consisting of one or more of the original developers, has responsibility for incorporating changes and suggesting things that need to be done. Modified versions of the source code are posted on the Internet and available for free to anyone who wants to use it or modify it further. Over time, users may end up running the software with other hardware/software combinations than did the original developers, identifying either problems that had originally escaped detection or worthwhile features to add. These users can provide feedback to the developers (or become developers themselves). Through this ongoing process the software becomes tested, debugged, and developed.

This approach differs from the commercial approach in many ways. First, there is typically little analysis of consumer needs other than introspection: "what would I like my software to do?" This may be augmented by user feedback, but these users are self-selected; except in unusual circumstances, they are not drawn randomly from the universe of potential users of the software. Second, there is little extensive, formal testing of the type that commercial firms often must engage in: internally test using hundreds, perhaps thousands of hardware/software configurations in a controlled manner. Testing is instead performed by the users who try versions of the software in uncontrolled environments, much like "beta" tests for commercial software developers (although perhaps with more sophisticated users providing feedback to the developers). Third, the development of open source software is less structured than is the development of proprietary software. Although the core developers may provide direction, changes in the software result much more from individual action. Some observers might consider this a benefit: "innovations" can come from anywhere. Others might consider it a potential hindrance: it may prove difficult to move a project forward on a coherent basis.

C. Incentives for Participating in Open-Source Projects

The incentives for writing open-source software are much different than for proprietary software. Understanding these incentives is important for making conjectures about the evolution of open source. There are two major economic possibilities: individuals choose to develop open-source software, essentially in their spare time; or firms pay programmers to develop open-source software. A non-economic possibility is that governments or universities choose to fund the development of open-source software; that is a policy issue that we discuss below.

1. Why Individuals Work on Open-Source Software

Why programmers donate time to open-source software projects is a subject that has generated considerable discussion.[87] Open-source advocates have suggested several motives, four of which involve non-financial rewards:

- It is fun. Since a programmer is free to pick and choose among open-source projects, he need only work on matters of interest.
- It is prestigious. Success at open-source development rates highly among those whose opinions the programmers most value—other programmers.
- It "scratches an itch."[88] Programmers attack problems that they personally face[89] or because they are intrigued by the intellectual challenge.
- It meets an ideological urge—the desire for free software and concern over Microsoft's "domination" of software.[90]

[87] See, for example, Josh Lerner and Jean Tirole, "Some Simple Economics of Open Source," *Journal of Industrial Economics*, Vol. 50:2, forthcoming June 2002; Justin Pappas Johnson, "Some Economics of Open Source Software," December 11, 2000 (presented at the IDEI Toulouse conference on "The Economics of the Software and Internet Industries," January 18–20, 2001); Eric Raymond, "The Magic Cauldron," August 25, 2000, http://www.tuxedo.org/~esr/writings/cathedral-bazaar/magic-cauldron/index.html (downloaded April 2, 2002).

[88] "Every good work of software starts by scratching a developer's personal itch." Eric Raymond, "The Cathedral and the Bazaar," http://www.tuxedo.org/~esr/writings/cathedral-bazaar/ (downloaded April 2, 2002)

[89] Of course, not all possible motivations apply in all circumstances. For example, considering the amount of effort that Linus Torvalds and others have devoted to developing Linux, it would have been much more efficient for them to have purchased commercial copies of Unix.

The "scratches an itch" motive has been considered by some analysts as leading to something like a cooperative of users. A number of developers all consider a particular type of software as potentially useful, so they pool their talents to develop the software. With this type of motivation, the GPL has sometimes been considered beneficial as an enforcement mechanism: it ensures that no one can take the collective intellectual property, add some private intellectual property, and treat the whole as a private good.

Motivations related, even indirectly, to financial rewards are generally dismissed by open-source advocates like Eric Raymond[91]—though Raymond does acknowledge that "It can get you a better job offer, or a consulting contract, or a book deal."[92] This is similar to the signaling incentive that Lerner and Tirole explore.[93] But according to Raymond, this sort of opportunity "is at best rare and marginal for most hackers."[94]

2. Business Models Based on Open Source

Unpaid volunteers are not the only possible source of labor for open source. Businesses may have incentives to "donate" labor to the development of open source, and in fact several have done so to some extent. These incentives depend on the extent to which funding an open-source software project stimulates the demand for other products or services sold by the firm.

(...continued)

[90] To gain an understanding of how some members of the open source community feel towards Microsoft, visit http://slashdot.org/, which regularly has anti-Microsoft posts, or read the "talk back" comments on articles about Microsoft posted at http://www.zdnet.com/.

[91] Eric Raymond is sometimes considered the resident anthropologist of the open source movement. ("An Interview with Raymond," January 24, 2001, http://opensource.oreilly.com/news/raymond_0101.html (downloaded April 2, 2002).) He has written extensively on subjects related to open source, such as the advantages he perceives from open source development and the motivations that programmers have to write open source software. For links to Raymond's writings, see http://www.tuxedo.org/~esr/writings/cathedral-bazaar/ (downloaded April 2, 2002). For Raymond's commentary on some internal Microsoft documents about open source (the "Halloween documents"), see http://www.opensource.org/halloween/ (downloaded April 2, 2002).

[92] Eric Raymond, "Homesteading the Noosphere," chapter 5, http://tuxedo.org/~esr/writings/homesteading/homesteading/homesteading.ps (downloaded April 2, 2002).

[93] Josh Lerner and Jean Tirole, "Some Simple Economics of Open Source," *Journal of Industrial Economics*, Vol. 50:2, forthcoming June 2002.

[94] Eric Raymond, "Homesteading the Noosphere," chapter 5, http://tuxedo.org/~esr/writings/homesteading/homesteading/homesteading.ps (downloaded April 2, 2002).

The circumstances under which a for-profit firm has incentives to invest in open-source, particularly under the GPL, may be limited. Consider two otherwise identical firms, where the first invests in developing open-source software and the second does not. Whatever open-source improvements the first firm comes up with (at least under the GPL) must be made quickly available to the second firm. So unless the process of open-source development creates parallel commercial opportunities, the first firm will necessarily have higher costs than the second. To put it another way, the first firm can expect to succeed only if open-source investments give it credibility unavailable to the second firm, a shorter lead time in developing commercial products or services, lower costs for the other products and services, or some other benefit that more than offsets the cost of devoting resources to open-source software.

Published articles on commercial firms and open-source software sometimes confuse the issues of ***using*** versus ***developing*** open source.[95] Any number of firms can have incentives to ***use*** open-source software. For example, a firm deciding what software to use for hosting a Web site might well consider the Apache Web server (open source but not GPL) as well as server software from Sun, Microsoft, and many other firms. Decisions about software use would normally be based on standard commercial considerations: features, stability, speed, ease of use, other characteristics, and price. For some firms and some software, the ability to modify the source code might be perceived as yet another useful characteristic. For other firms and other software, the availability of the source code would provide few if any benefits.

The issues are much different, however, when it comes to incentives for firms to ***develop*** open-source software, particularly GPL software. Here we consider three major ways in which firms may find that developing open-source software helps them sell complements: proprietary software, hardware, and services.

[95] For example, "With a column called The Open Source, not a week goes by without someone asking me how anyone makes money when they give away the source code for their programs. … Using open source to make money is a no-brainer. (Nicholas Petreley, "Open-Source Economics," *InfoWorld*, August 24, 2001, available at http://staging.infoworld.com/articles/op/xml/01/08/27/010827oppetreley.xml?Template=/storypages/printfriendly.html (downloaded May 20, 2002))." (Nicholas Petreley, "Open-Source Economics," *InfoWorld*, August 24, 2001, available at http://staging.infoworld.com/articles/op/xml/01/08/27/010827oppetreley.xml?Template=/storypages/printfriendly.html (downloaded May 20, 2002)).

Although some open source advocates have been bullish about the potential for firms to build profitable businesses around the development of open source, others are more skeptical (and the more purely ideologically motivated simply do not care). For example, Bruce Perens, primary author of "The Open Source Definition,"[96] recently stated:

> [Michael Robertson of Lindows.com] also commented about the lack of successful Linux companies. This is not due to the community treatment of Linux businesses, but the fact that Open Source is not a business and should not be treated as one. It's successful when operated as a cost-center, in businesses that make their money some other way. The most successful ones use the software they develop for some business purpose: for example, Apache developers use the software to implement web sites for their business, IBM and HP make money by selling hardware that runs with Linux, not by selling Linux. Eric Raymond and others theorized that support would be a good way to fund Open Source, but the support model has under-performed so far, because the early adopters are too self-supporting. Sales of proprietary software to support the Open Source development are also underperforming, as Linux customers, even within the Fortune 500, have become wary of dependence on non-Open-Source. Thus, no Linux distribution has been more than marginally profitable so far. My surmise is that over the long term a non-profit like Debian supported by hardware manufacturers and other businesses will work best.[97]

a. Sell Complementary Software

Under this approach, businesses use open-source software to help promote proprietary software, subscriptions, or advertising revenue from a Web portal site (such as www.msn.com, www.netcenter.com, or www.yahoo.com).[98] For example, a business could promote open-source client (i.e., desktop or laptop) software to help sell proprietary server software or to drive traffic to a portal site. "Complementary" software might be a specialized version of the open-source software or proprietary software that works in conjunction with the open-source software. For the complementary software strategy to work, the profits made possible on the

[96] "The Open Source Definition: Version 1.9," http://www.opensource.org/docs/definition.html (downloaded April 16, 2002).

[97] Bruce Perens, "Open Letter to Michael Robertson," April 13, 2002, http://www.lwn.net/daily/perens-robertson.php3 (downloaded April 16, 2002).

[98] Eric Raymond, "The Magic Cauldron: Indirect Sale-Value Models," http://www.tuxedo.org/~esr/writings/cathedral-bazaar/magic-cauldron/x227.html (downloaded April 2, 2002).

complementary commercial product must be sufficient to compensate for the cost and risk of the investment in the open-source project.

Several firms have supported the development of GPL software in conjunction with their own proprietary software. Given the ideological issues surrounding the GPL, there is a tension inherent in this business model: how can a firm that wants to market proprietary software support the GPL, which is designed to drive proprietary software out of use? But some firms do. In addition to IBM, discussed below under hardware vendors, the most public of the firms involved in developing GPL software have included Corel, Ximian, MySQL, and theKompany.com.[99]

Corel came out with its own Linux distribution, developing installation and other tools to make it easy to install. It also provided some support to the WINE project[100] in the course of adapting (or "porting") its WordPerfect Suite to run on Linux. Corel has since shed its Linux work and has discontinued development of the Linux version of its office suite.[101] This effort must be tagged as a failure. Just as Perens explained in the quote above, it seems that Corel was unable to make money by selling Linux; it did no better in trying to sell WordPerfect for Linux.

Ximian has developed a GPL program called Evolution that runs on Linux;[102] it has also been a major contributor to other open source projects, including GNOME, a GPL "desktop"

---

[99] Several firms have supported the development of non-GPL open-source software, typically attempting to license an enhanced version of the software under proprietary terms.

[100] The WINE project is an attempt to clone the Windows APIs so that applications designed for Windows will run on Unix-like operating systems. The WINE project was originally licensed with a BSD-style license, but it recently switched to the LGPL, another license that is promoted by the FSF, but not as enthusiastically because it is not as restrictive as the GPL. (Jeff Tranter, "Frequently Asked Questions About Wine and Corel," http://linux.corel.com/support/wine_faq.htm (downloaded April 5, 2002); Alexandre Julliard, "Conclusions," March 4, 2002, http://www.winehq.com/hypermail/wine-license/2002/03/0029.html (downloaded April 5, 2002); Free Software Foundation, "Why You Shouldn't Use the Library GPL for Your Next Library," http://www.fsf.org/philosophy/why-not-lgpl.html (downloaded April 2, 2002).)

[101] The latest version of its office suite for Linux is WordPerfect Office 2000; the latest version for Windows is WordPerfect Office 2002. ("Corel Linux Community: Products," http://linux.corel.com/products/ (downloaded April 5, 2002); "WordPerfect Office 2002 Standard," http://www3.corel.com/cgi-bin/gx.cgi/AppLogic+FTContentServer?GXHC_GX_jst=5a964475662d6165&GXHC_gx_session_id_FutureTenseContentServer=355c17803d3e054a&pagename=Corel/Product/Highlight&id=CC1MSNDMYJC&highlight=requirements (downloaded April 5, 2002); Xandros Press Release, "Xandros Announces Strategic Licensing Agreement with Corel Corporation," August 29, 2001, http://www.xandros.net/news_full.html (downloaded April 5, 2002)).

[102] "Ximian Evolution 1.0," http://www.ximian.com/products/ximian_evolution/ (downloaded April 5, 2002).

user interface for Linux and other Unix-like operating systems. Evolution mimics the operation of Microsoft Outlook, which provides e-mail and other capabilities, some of which are available only when used in conjunction with Microsoft's own e-mail server. Ximian offers a proprietary program that will let Evolution work with Microsoft's e-mail server, thereby gaining the same types of capabilities on Linux that are available to users of Outlook on Windows.[103] This product is too new to evaluate as a success or failure.

theKompany.com has also developed both GPL and proprietary products. For example, it developed a specialized graphics program, Kivio (which has capabilities similar to Microsoft's Visio). Kivio ships under the GPL as part of the KOffice suite for the KDE desktop for Unix-like operating systems. The firm has announced two ways it hopes to make money from this: offering proprietary add-ons to this program; and (soon) offering an enhanced version of the program that runs not only with KDE but also with Windows.[104] The firm also has announced, however, that it will not use the GPL for future products.[105] Its use of the GPL should therefore be classified as a long-run failure.

MySQL AB is a firm whose major product, a database, is licensed (primarily) under the GPL.[106] In part, MySQL earns revenues by providing support services to users of its database (services that in principle could be provided by anyone else who wanted to learn enough about the product). But MySQL also earns revenues from non-GPL licensing of its flagship product, for firms that want to combine their software with MySQL's in ways that would not be permitted under the GPL. To date, MySQL appears to be successful with its mixed strategy.[107,108]

---

[103] "Ximian Connector for Microsoft Exchange," http://www.ximian.com/products/connector/faq.html (downloaded April 5, 2002).

[104] "Kivio mp," http://www.thekompany.com/products/kivio/ (downloaded April 2, 2002).

[105] Shawn Gordon, "Running a Corporation in an Open Source World," March 18, 2002, http://www.linuxandmain.com/essay/sgordon.html (downloaded April 5, 2002).

[106] Earlier versions of MySQL were released under an even more restrictive license than the GPL. http://www.mysql.com/support/arrangements/mypl.html ("MySQL Free Public License Version 4," March 5, 1995, http://www.mysql.com/support/arrangements/mypl.html (downloaded April 5, 2002).)

[107] MySQL AB Press Release, "Record Sales, New Financing Fuel Growth at MySQL," April 22, 2002, http://www.mysql.com/news/article-96.html (downloaded May 20, 2002).

b. Sell Complementary Hardware

Investing in open-source software could increase the demand for hardware or cut the costs of producing complementary software that is bundled with the hardware. IBM is perhaps the most public "success" story for a hardware firm supporting the development of open-source software, some (not all) of which is under the GPL.[109] IBM claims to have invested $1 billion (including marketing expenditures) for a variety of open-source initiatives, including adapting Linux (GPL) and Apache (not GPL) to IBM's various computer hardware platforms.[110] IBM's hardware business is unusual in that it markets several fundamentally different types of servers with mutually incompatible operating systems. Linux permits IBM to unify its server product line, so that proprietary IBM software (and other software) can be used on all the different servers. IBM is also unusual for a hardware company in that it is also one of the largest software vendors in the world.[111] IBM uses Linux (and other open-source software) to help promote its sales of its hardware as well as its proprietary software. We are aware of no other firms in a situation like IBM's.

Intel has supported work on Linux, in part through the provision of hardware and other resources to developers. Intel has a clear interest in wanting high-quality operating systems that run well on computers powered by its processors. To encourage the development of software that takes full advantage of its processors, Intel provides a variety of software, including system

---

(...continued)

[108] A similar dual licensing strategy is followed on a lesser scale by the author of an advanced file system for Linux. "This project is GPL'd, but I sell exceptions to the GPL to commercial OS vendors and file server vendors. It is not usable to them without such exceptions…" Hans Reiser, "ReiserFS v.3 Whitepaper," http://www.reiserfs.org/content_table.html (downloaded April 8, 2002).

[109] For example, IBM released source code to its Andrew File System under the IBM Public License and released source code to its Journaled File System for Linux under the GPL ("Open AFS Overview," http://www-124.ibm.com/developerworks/oss/afs/info.html (downloaded April 5, 2002); "Journaled File System Technology for Linux," http://www-124.ibm.com/developerworks/oss/jfs/index.html (downloaded April 5, 2002)).

[110] James Evans, "IBM to invest almost $1 billion in Linux development," *InfoWorld*, December 12, 2000, available at http://www.infoworld.com/articles/hn/xml/00/12/12/001212hnibmlin.xml (downloaded April 3, 2002).

[111] Its more popular server-based products include DB2 (a database) and Domino (the server component of its Notes communications/groupware product); both products are available for Linux. See "DB2 for Linux," http://www-3.ibm.com/software/data/db2/linux/ (downloaded May 16, 2002); "Lotus Domino," http://www.lotus.com/home.nsf/welcome/domino (downloaded May 16, 2002).

libraries, test suites, programming tools, and experimental code.[112] We know of no other major hardware vendors that have publicly announced major initiatives to encourage the development of open-source software, although some hardware vendors, such as Hewlett Packard, have developed Linux drivers for their hardware.[113] In addition, Hewlett Packard has recently supported the development of open source kernel extensions such as openMosix for the IA64 microprocessor family.[114] And Hewlett Packard also "sponsors" or is a member of a variety of open source organizations.[115] Some other vendors have announced support for open source in more modest ways.[116]

There are some conflicts between the goals of profit-oriented hardware companies and the open-source movement. Hardware companies want to differentiate themselves from each other, and software is one dimension on which they have tried to do that. That is what the "Unix wars" of the 1980s were about: competing hardware vendors (IBM, Sun, SGI, Digital, Hewlett-Packard, and others) offered proprietary (and, by design, mutually incompatible) versions of Unix to differentiate their products. Open-source advocates, and especially those responsible for Linux, want to prevent open-source code from "forking" into mutually incompatible versions in the way Unix did. But the widespread use of a standard version of Linux might reduce barriers to entry into the manufacturing of high-end computers much as the widespread use of a proprietary standard (MS-DOS and then Windows) reduced barriers to entry into personal computers. That would hurt IBM and either reduce IBM's incentives to

[112] See "Open Source from Intel," http://developer.intel.com/software/products/opensource/ (downloaded April 5, 2002).

[113] Hardware vendors are sometimes reluctant to develop open source drivers, because doing so can make it easier for competitors to improve their own drivers.

[114] Moshe Bar, "Porting Linux to the IA64 Platform," Byte.com, May 13, 2002, http://www.byte.com/documents/s=7182/byt1021067742738/0513_moshe.html (downloaded May 14, 2002).

[115] "HP and Open Source," http://www.hp.com/united-states/linux/opensource.html (downloaded April 5, 2002).

[116] For example, in August 2000 companies including Compaq HP, IBM, and Sun "announced their support" for the GNOME Foundation, which develops the GNOME graphical desktop (GNOME Foundation Press Release, "The GNOME and Linux Communities and Industry Leaders Join to Create the GNOME Foundation," August 15, 2000, http://www.gnome.org/pr-foundation.html (downloaded May 16, 2002)). Sun has continued to work with the GNOME Foundation, and recently contributed to GNOME's Accessibility Framework (GNOME Foundation Press Release, "Making GNOME Accessible—Opening New Doors at the Workplace for Users with Disabilities," August 28, 2001, http://www.gnome.org/pr-accessible.html (downloaded May 16, 2002)).

invest in Linux or encourage IBM to figure out ways to differentiate the flavor of Linux that ran on its machines.

c. Sell Complementary Services

Firms can attempt to make money from developing open-source software by selling associated services. Although a firm would not necessarily need to invest in paying programmers to write open source code, by doing so it may obtain specialized knowledge about open source that might help it provide better services. Or it might obtain credibility about open source that would make it better able to market its services. Several companies have followed this approach to various degrees. Cygnus Solutions was heavily involved in developing and maintaining the open-source GNU compilers. It sold support for these products, and it contracted with hardware firms to develop versions of the compilers for new chips. It is possible that Cygnus's familiarity with these compilers gave it a cost advantage over other firms in performing these tasks. Cygnus Solutions was purchased by Red Hat, a Linux distribution company, in 1999.[117]

Red Hat and other Linux distributors are arguably selling a service, namely "the value added by assembling and testing a running operating system that is warranted (if only implicitly) to be merchantable and to be plug-compatible with other operating systems carrying the same brand."[118] However, there appear to be few barriers to entry to such a business, which makes it difficult to make a profit while also funding the development of Linux or other open-source products. This is particularly true when other firms can (and do) look at what Red Hat chooses to distribute, and then distribute the same code, packaged in the same way.[119] Several

[117] "Red Hat to Acquire Cygnus and Create Global Open Source Powerhouse," November 15, 1999, http://www.redhat.com/about/presscenter/cygnus_1999/redhat-cygnus111599.html (downloaded April 2, 2002).

[118] Eric Raymond, "The Magic Cauldron: Indirect Sale-Value Models," http://www.tuxedo.org/~esr/writings/cathedral-bazaar/magic-cauldron/x227.html (downloaded May 15, 2001).

[119] A review of one Linux distribution, Mandrake 8.0, reported that potential users could download the product for free, purchase the product for $69.95 from Mandrake, or purchase a CD with the product from another party for $4.95 ("Linux Mandrake 8.0," http://www.thedukeofurl.org/reviews/misc/mandrake80/printable.shtml (downloaded June 5, 2001).)

Linux distributions have reported rocky financial results within the last year, and even Red Hat cannot be considered a resounding financial success.[120]

Several firms that support open source have announced plans to offer automatic software updating services in connection with their open-source work: Red Hat, Eazel, Caldera, and Ximian. Red Hat and Ximian were mentioned above.[121] Eazel, which developed a file management tool under the GPL, no longer exists.[122] Caldera, a Linux vendor that is not entirely happy with the GPL and sells proprietary software as well as Linux, offers a similar service.[123] Entry into an industry like this does not seem particularly difficult.

D. The Performance of Open Source

Open-source production methods have created several software products that are used widely in their relevant categories. After reviewing these successes, we summarize the status of open source projects under development. An understanding of where and why open source has succeeded and failed is needed to evaluate policies designed to promote open source.

1. Successes and Failures of Open -Source Software

It is useful to divide the successes into those distributed under the GPL and those under BSD-style licenses.

---

[120] For example, see Mandrake ("Mandrake Linux Users Club Answers," April 4, 2002, http://www.mandrakelinux.com/en/club/club-answers.php3 (downloaded April 5, 2002)) and SuSE (Stephen Shankland, "SuSE Linux Gets New CEO, Cuts Staff," *CNET News.com*, July 23, 2001, available at http://news.com.com/2102-1001-270372.html (downloaded April 5, 2002)). Red Hat projected that it would lose $140 million over fiscal year 2002 on $79 million in total revenue. ("Red Hat, Inc. Consolidated Statements of Operations—FY2002," December 31, 2001, http://media.corporate-ir.net/media_files/NSD/RHAT/reports/GAAP_Statement_Ops_FY2002.pdf (downloaded April 5, 2002).)

[121] See "Red Hat Network: Overview of Services," https://rhn.redhat.com/feature_comparison.pxt (downloaded April 5, 2002); "Ximian Red Carpet: Automated Software Maintenance and Version Management," http://www.ximian.com/products/ximian_red_carpet/ (downloaded April 5, 2002).

[122] "Linux Specialist Eazel Calls It Quits," http://www.zdnet.com/eweek/stories/general/0,11011,2761184,00.html, downloaded May 30, 2001; "Eazel's Demise Is Official," http://linuxtoday.com/mailprint.php3?action=pv&ltsn=2001-05-16-004-20-NW-GN, downloaded May 30, 2001.

[123] Mary Foley, "Caldera CEO: MS Right to Shun Open Source," *eWEEK*, May 8, 2001, available at http://zdnet.com.com/2102-11-503692.html (downloaded April 5, 2002); "Caldera Volution Online," http://www.caldera.com/products/volutiononline/ (downloaded April 5, 2002).

a. GPL Successes

Probably the most famous GPL software is Linux, an operating system widely popular on server computers but, at least relatively, much less so on client computers.[124] Also widely praised and used are the MySQL database and Samba, software that lets servers running Linux and other Unix-like operating systems emulate Windows servers.[125] "Clustering" software for Linux, creating relatively inexpensive "supercomputers," might also be judged a success.[126] The development tools from the GNU project (such as the Emacs editor and the GCC compilers) probably are not as widely used as many commercial development tools, but they are widely used by open-source developers.

The GNOME and KDE desktops for Linux and other Unix-like operating systems should probably be considered qualified successes. They provide graphical user interfaces for these operating systems, somewhat like those for Windows and the Macintosh but not yet as user-friendly. Linux's relative lack of success on client computers means that these desktops are not yet widely used, but they may have promise.

b. BSD-Style Successes

The list of famous successes under BSD-style licenses seems longer than those under the GPL. The BSD family of Unix operating systems has a long history—the first computers from Sun were based on it, as were many others. The most popular open source versions of

---

[124] In absolute numbers, Linux is used on more clients than servers. In 2001, the installed base of Linux was estimated to be about 6.6 million units on clients and 4.0 million units on servers. IDC Report #26952, "Worldwide Linux Operating Environments Forecast, 2002–2006: Client Shipments Pick Up the Pace" February 2002, Table 4, p. 8.

[125] MySQL AB Press Release, "Leading European VCs Invest in MySQL AB," November 12, 2001, http://www.mysql.com/news/article-84.html (downloaded May 20, 2002); Michael Vizard and Steve Gillmor, "CEO of MySQL Explains Why Open-Source May Be Right Approach for Many Enterprises," *InfoWorld*, December 20, 2001, http://staging.infoworld.com/articles/hn/xml/01/12/20/011220hnmickos.xml?Template=/storypages/printfriendly.html (downloaded May 20, 2002); Samba has been praised on various occasions and was being used by thousands of users in mid-1997; its current number of users is surely much higher ("Samba Survey," May 5, 1997, http://www.samba.org/samba/survey/ (downloaded May 20, 2002); "i3 Awards Finalists," *eWEEK*, April 29, 2002, available at http://www.eweek.com/print_article/0,3668,a=26153,00.asp (downloaded May 20, 2002)).

[126] Rick Cook, "Supercomputers on the Cheap," *LinuxWorld*, April 13, 2000, available at http://www.cnn.com/2000/TECH/computing/04/13/cheap.super.idg/ (downloaded April 5, 2002).

BSD Unix currently available (FreeBSD, OpenBSD, and NetBSD) do not seem to be as popular as Linux, with one partial exception: Apple's latest operating system (Mac OS X) is based on a version of FreeBSD. Apple's user interface is not open source, but the BSD Unix that underlies it is open source.[127] Two other key pieces of software came out of work done at Berkeley roughly two decades ago: BIND and Sendmail. Current versions of BIND are widely used on servers for resolving Internet addresses, converting Web addresses (like www.nera.com) to IP addresses (like 64.23.37.68).[128] Sendmail was the first modern e-mail server software and is still widely used.[129]

Other widely-used software also has BSD-style licenses: Apache is perhaps the most widely used Web server software; Xfree86 is an implementation of a windowing system for Linux and other Unix-like operating systems for Intel-compatible computers—it seems to be included with essentially all Linux distributions, of use on both clients and servers; and Perl is a scripting language widely used on servers and on clients by developers.[130]

c. Some Failures

By "failures" we mean categories of failures for open source as a whole, not individual product failures. In general, open source has not been successful at developing user-friendly software aimed at mass-market users. To date, open-source office suites have not succeeded. Office suites are under development in conjunction with both the KDE and GNOME desktops, but neither can yet be considered a success.[131] The best of the existing open-source office suites

[127] "Mac OS X System Architecture," 2001, http://developer.apple.com/macosx/architecture/index.html (downloaded April 5, 2002).

[128] "Internet Software Consortium BIND," http://www.isc.org/products/BIND/ (downloaded April 5, 2002).

[129] "Sendmail—Company Overview," http://store.sendmail.com/cgi-bin/smistore/company.jsp?BV_UseBVCookie=Yes&filepath=overview/index.shtml&heading=Company%20Overview (downloaded March 30, 2002).

[130] Unlike many programming languages, a "scripting language" is not normally compiled. It is often used to manipulate files on a computer, not to develop major programs such as a word processor.

[131] GNOME Office is a different kind of office "suite." Rather than consist of a specific set of applications, it considers itself to include any GPL applications that use a common component architecture (for linking files) and a common set of programming libraries. ("GNOME Office," http://www.gnome.org/gnome-office/index.shtml (downloaded April 2, 2002).)

is probably OpenOffice[132]—which is based on the source code for a proprietary product, StarOffice, whose use was too tiny for market research firms to measure when Sun acquired its developer.[133] We are unaware of any commercial-quality games developed under the GPL.

2. Open Source Projects under Development

The number of open-source projects currently under development is very large. Most of these seem to rely on the GPL rather than licenses with fewer restrictions on commercialization. SourceForge is a Web site that provides free hosting services to open-source projects. In early May, 2002, we downloaded from the SourceForge Web site information on 38,610 projects that are hosted there. Of these, 25,194 projects provided information on the licenses used (some projects use multiple licenses). Of these, 18,133 used the GPL at least in part, and 20,220 used either the GPL or the LGPL (a cousin to the GPL, also favored by the FSF for some purposes). The GPL therefore accounts for 47 percent of the projects hosted at SourceForge and 72 percent of the projects with a known license type. Combined, the GPL and LGPL account for 80 percent of the projects with a known license type. The reasons for the popularity of the GPL are not entirely clear, although two possible explanations were mentioned above: the FSF and related highly vocal supporters of non-proprietary software tend to favor the GPL for ideological reasons; and the GPL might be viewed as enforcement mechanism for the cooperative sharing of programming effort.

SourceForge provides other useful breakdowns of the projects they host. Of the 25,852 projects whose development status is known, only 4,712 (18 percent) had reached either the

---

[132] A recent review in the *Washington Post* states, "After using the Windows version of OpenOffice for the past week and a half, I can attest that it either matches or beats Microsoft Office in features and ease of use, at the cost of slower performance on older computers and the occasional slight garbling of complicated Microsoft Office documents. It's hardly perfect, but somebody in Redmond ought to be worried about this program" (Rob Pegoraro, "The Office Suite that Lets You See Past Redmond," ***Washington Post***, May 12, 2002, p. H07, available at http://www.washingtonpost.com/ac2/wp-dyn/A4246-2002May11?language=printer (downloaded May 16, 2002)).

[133] After Sun acquired the developer of StarOffice, it made the product available for free and made the source code available to what became the OpenOffice project. Sun has announced that it will begin charging for the next version of StarOffice, but OpenOffice remains an open source project. Sun Microsystems Press Release, "Sun Expands StarOffice Software Offering," March 19, 2002, http://www.sun.com/smi/Press/sunflash/2002-03/sunflash.20020319.1.html (downloaded April 24, 2002).

"production" or the "mature" stages.[134] The rest were in various planning or development stages. The number of projects with information on "intended audience" came to 25,402; of these, only 55 percent included "end users/desktop" among the intended audience (again, projects could have multiple intended audiences). In contrast, 75 percent had developers or system administrators among the intended audience.

Only tentative conclusions can be drawn from numbers such as these. It seems clear, however, that for the huge number of currently active open-source projects, the GPL is by far the most popular license. It also seems clear that end users are not the main audience of open-source developers.

E. Summary

Open-source software has succeeded in some areas but not others. The lack of property rights in open-source software means that, except in special circumstances, firms have little or no incentive to devote substantial resources to the development of open source. Firms that concentrate on the distribution of open source software (such as Red Hat and other Linux distributors) are in an industry with essentially free entry; "buying" more software from them is likely to have little or no effect on the supply of programmers to develop open-source software.

V. Comparisons of Proprietary and Open-Source Software

Some government policy proposals have been based on claimed advantages of open-source software over proprietary software. We compare open-source and proprietary software in two ways in this section. Part A examines the advantages and disadvantages of both approaches. Part B examines an oft-made claim that the open-source approach results in more innovation. Part C considers the evolution of open-source and proprietary software under government neutrality.

[134] Projects could be in multiple stages, such as a "production" stage for one version and a "beta" stage for another. These figures count a project as in the "production" or "mature" stages if either of those stages had been reached by the project.

A. Advantages and Disadvantages of Each Approach

The relative advantages and disadvantages of open-source and proprietary software are of course mirror images: what is an advantage for open source is a disadvantage for proprietary software; what is an advantage of proprietary software is disadvantage for open source.

1. Open Source

a. Advantages

Open-source software in general, to some extent including GPL software, has several strengths. One involves the use (as opposed to the creation) of intellectual property. Intellectual property may be expensive or difficult to create, but, once created, the marginal cost of using it is zero. As a result, for any already-created piece of intellectual property, society benefits most when that intellectual property is made available to all for free. Open source more or less does so, with the GPL in this regard being less attractive than other open-source licenses to commercial firms. Using code licensed under the GPL can impose costs that using code available under other licenses does not—the inability to mix that program code with proprietary program code.

The availability of source code for open-source programs means that technically adept users can tailor the software to their particular needs. They can also fix bugs that they come across and provide those fixes to other users. These advantages will appeal more to business users than to typical home users, of course.

Since source code is available for technically adept users to inspect, it is possible that bug fixes may become generally available to all users (not just the technically adept) more quickly than with proprietary software; whether such bug fixes can easily be put into the hands of general users is less clear.

At least in theory, open source may be more protective of "privacy" than is proprietary software: with open source, it would be difficult for a programmer to include code that would somehow "spy" on unsuspecting users. Other programmers could simply remove such code. Whether this theoretical advantage is a real-world advantage is not clear.

b. Disadvantages

Because the source code is open to one and all, open-source developers have limited opportunities to earn a pecuniary return on the time and effort that they invest in their work. As discussed above, non-pecuniary rewards can certainly provide some motivation, but we know of no reason that non-pecuniary rewards should be more important in software development than in other fields. The limited pecuniary rewards available to open-source developers will tend to limit the supply of effort devoted to these activities.

Firms likewise have limited opportunities to earn pecuniary rewards for their investments in open-source projects. As a result, they have limited incentives to perform the types of consumer research, into usability and consumer needs, that developers of proprietary firms do; they have no easy way to earn a return on any such investments.

The development process, which primarily involves interactions among the technically adept, provides few incentives to identify and eliminate issues that might be problems for the less technically inclined.

Open source is also subject to what is sometimes call "fragmentation"—the creation of multiple incompatible versions of the same software. Different Linux distributions, for example, have taken different approaches to organizing where operating system and other files are stored. This makes it difficult for developers of applications for Linux to develop installation routines: a routine that works for one Linux distribution will not work on another.[135] Linux has not yet fragmented to the same extent that Unix did, and perhaps it never will; but differences to date across Linux distributions have posed problems to developers and non-technical users.[136]

---

[135] The Linux Standard Base initiative is trying to overcome this particular problem. ("Linux Standard Base," http://www.linuxbase.org/ (downloaded April 2, 2002)).

[136] In addition to the problem with file locations, different Linux distributions have sometimes included different (and incompatible) program "libraries" used in running applications (Andrew Leonard, "Is Red Hat becoming Linux's Microsoft?" *Salon.com Technology*, July 14, 1999, available at http://www.salon.com/tech/feature/1999/07/14/redhat/index1.html (downloaded January 21, 2002); Charles Babcock, "Linux vs. Linux" *Inter@ctive Week*, March 20, 2000, available at http://www.zdnet.com/filters/printerfriendly/0,6061,2470425-2,00.html (downloaded January 21, 2002)).

2. Proprietary Software

a. Advantages

As with other industries based on intellectual property, the possibility of earning returns on an investment encourages firms to make the investment in the first place. For example, one of the big movies of 2001 was the first installment of *Lord Of The Rings*. The makers of this movie would have had no incentive to devote as much time, effort, and money into the script, stars, and special effects if everyone who saw the movie had the right to give away copies to everyone else. Software is no different in that regard. One might perhaps argue about the extent of intellectual property protection that should be given to computer software, but arguing that no protection should be available is an untenable position from an economic perspective.

With proprietary software, a firm can control the destiny of its products. It has profit motives to make its products highly valued by consumers. As a result, vendors typically try to make their products backwards compatible with earlier versions, so that documents and user training can experience a smooth transition from an older version to a newer one.

Related is the issue of fragmentation. One of the strengths of Windows and the Macintosh is that they provide consistent platforms on which applications can be run.[137] Developers for Windows (or the Macintosh) know that if they write their programs in certain ways, these programs should run on all computers that meet specific hardware and operating system requirements (e.g., Windows ME, Windows 2000, or later; OS X or later). With the free-for-all customization possible for Linux, that is not true for Linux developers. A developer who wants to write a program that permits linking of objects (such as putting a graph from a spreadsheet into a word processing document) can do so readily on either Windows or the Macintosh; someone wanting to do so for Linux faces an uphill battle.[138]

---

[137] This consistency is important in providing the benefits of network effects to users.

[138] Both the KDE and GNOME desktops have "component" models—but they are different. A program written to use the KDE component model cannot interoperate with a program written to use the GNOME component model (Stephen Shankland, "The Struggle for the Future of Linux," *CNET News.com*, February 26, 2001, available at http://news.com.com/2102-1082-253153.html (downloaded April 5, 2002)). The use of Linux in "embedded" devices (such as cell phones, handheld computers, robotics) faces similar problems (Matthew Broersma, "Embedded Linux Crying Out for Standards," *ZDNet UK*, May 17, 2002, available at http://zdnet.com.com/2102-1104-916331.html (downloaded May 20, 2002)).

b. Disadvantages

A technically adept user who encounters a bug in a proprietary program cannot fix it himself; the most he can do is report the bug to the vendor and hope a fix is made available soon (but the vendor of a proprietary program has financial incentives to make it easy for users to obtain and install bug fixes). Similarly, a technically adept user cannot customize a proprietary program except in whatever ways the vendor has chosen to make the program extensible. Neither of these issues is important to the vast majority of home users, but they can be important to large customers. Even if the program vendor eventually releases a version with features more to the liking of a given user, the technically adept user might have been able to implement those features more quickly on his own.

B. Open Source: Innovation and Imitation

Discussions of "innovation" have sometimes debated whether the term should refer to the invention of technology or the popularization of technology.[139] By either measure, open-source software seems to have performed poorly. Although there are important exceptions, discussed below, the open-source movement has focused mainly on developing software that imitates successful proprietary software. Some of the early efforts (discussed below) under BSD-style licenses seem to have been truly innovative: BIND, the BSD family of operating systems, Sendmail, perhaps Perl, and the X Window system and its Xfree86 implementation for Intel-compatible computers. Apache (another BSD-style license) came a bit later but also has some claim to being innovative.

On the other hand, some of the early efforts from the FSF (now often licensed under the GPL), as well as many modern projects, seem more derivative and less innovative. One original objective of the FSF was to develop a "free" version of Unix—an imitative endeavor. The FSF work began with development tools—also imitative. The Linux kernel (leading to the numerous Linux distributions available today) was likewise an attempt to imitate the functionality of Unix. The GNOME and KDE desktops are not clones of the user interfaces of

[139] See, e.g., Marco Iansiti and Josh Lerner, "Evidence Regarding Microsoft and Innovation," AEI-Brookings Related Publication 02-4, April 2002, available at http://www.aei.brookings.org/publications/related/innovation.pdf (downloaded April 16, 2002).

either Windows or the Macintosh, but they attempt to bring the same kind of usability to Unix-like operating system. Samba is an explicit attempt to clone the functionality of Windows servers for use with Linux and Unix servers. MySQL is an implementation of a database with standard capabilities (called SQL).

Some observers have suggested that the open-source method tends to promote innovation more than the proprietary method—that the development method itself promotes innovation. For example:

> The development model encourages tremendous innovation. When developers can see and modify source code, they receive rapid feedback and a constant flow of ideas from other developers. [140]

Other factors are also clearly at work, however—copying and cloning is easier than innovating. For example, a recent interview with a key open-source developer (Miguel de Icaza) discussed a recent attempt (called Mono) to clone new computer language technology from Microsoft (part of what Microsoft calls .Net):

> He found a certain utility in the specification that lies at the center of .Net. That, coupled with the sentiment "It's a lot easier to implement than to design" an architecture, led Miguel and friends to start the Mono project.[141]

Lawrence Lessig has argued that open source contributes to an intellectual "commons," which can serve as a springboard toward further development.[142] He considers several open-source projects to constitute the "soul" of the Internet: Linux (an operating system), Apache (a Web server), Perl (a scripting language[143]), BIND (domain name software), and Sendmail (an e-mail server):

---

[140] "Open Source Pioneers Meet in Historic Summit," April 18, 1998, http://press.oreilly.com/opensource.html (downloaded April 2, 2002).

[141] Russell Pavlicek, "Get Mono from .Net?," *InfoWorld*, March 29, 2002, available at http://staging.infoworld.com/articles/op/xml/02/04/01/020401opsource.xml?Template=/storypages/printfriendly.html (downloaded April 16, 2002).

[142] This view is not unlike the FSF's view of a GPL "club," mentioned above. "This free code builds a commons." Lawrence Lessig, *The Future of Ideas*, Random House: New York, 2001, p. 57

[143] Unlike many programming languages, a "scripting language" is not normally compiled. It is often used to manipulate files on a computer, not to develop major programs such as a word processor.

> Together with the public domain protocols that define the Internet … this free code built the Internet. This is not a single program or a single operating system. The core of the Internet was the collection of code built outside the proprietary model.[144]
>
> Not strong, perfect control by proprietary vendors, but open and free protocols, as well as open and free software that ran on top of those protocols: these produced the Net.[145]

This is not quite right. It is true that the Internet has been built on open and free protocols—but open protocols are conceptually very different from open-source software.[146] And it is true that the Internet grew up around some of the products that Lessig considers its "soul," particularly BIND and Sendmail. But what was the direction of causality? The Internet was not made available for commercial purposes until 1991, when the U.S. National Science Foundation removed restrictions on commercial use of NSFNET.[147] It was originally developed under government sponsorship, connecting computer groups at universities and other research institutions. Until the Internet became commercial, there was little reason for firms to attempt to write proprietary software for it. The early, non-commercial Internet more resembled the mainframe era of the 1960s, with software written by its own, technically adept, users because otherwise there would be no software, than the commercial software era that roared into action in the 1980s with the widespread adoption of PCs and the development of corporate networks of PCs.

Of the five "soul" projects, only one was released under the GPL—Linux. The other products are all available under less commercially restrictive licenses.[148] How "innovative" are these projects? As economists, we do not pretend to provide firm answers to these questions.

---

[144] Lawrence Lessig, *The Future of Ideas*, Random House: New York, 2001, p. 56.

[145] Lawrence Lessig, *The Future of Ideas*, Random House: New York, 2001, p. 57. Lessig also quotes similar sentiments from Alan Cox, "second only to Linus Torvalds in the Linux chain."

[146] A "protocol" can be thought of as a definition of how to do something. Open protocols can be implemented in either open-source or proprietary software.

[147] "The Internet," http://www.nsf.gov/od/lpa/nsf50/nsfoutreach/htm/n50_z2/pages_z3/28_pg.htm#answer3 (downloaded April 16, 2002).

[148] Perl is available under two licenses, one of which is the GPL and the other of which is more permissive than the GPL. (Free Software Foundation, "Various Licenses and Comments about Them," http://www.gnu.org/licenses/license-list.html (downloaded April 16, 2002).)

As noted above, Linux began life explicitly as an attempt to clone Unix. The quality of Linux may well be high, but it seems imitative, not innovative.[149] Our understanding is that BIND helped make the Internet possible,[150] and that Sendmail ushered in the switch from older Internet e-mail to more modern Internet e-mail.[151] Perl was not the first scripting language,[152] but it does seem to have gained rapid acceptance and is widely used for some purposes. Apache was not the first Web server, nor is it the only Web server;[153] Web servers from Microsoft, Sun, and other companies are also widely used. On the other had, it is directly descended from "patches" (hence the name) to the early NCSA Web server, whose development had stalled.[154] On balance, its heritage and wide popularity suggest it should be considered innovative.

These four products—all of which are available under BSD-style licenses—have stronger claims to be innovative than does Linux.

Of course, pointing out a few products that may be innovative, in contrast with a few that do not, proves little. What should be considered innovative? Consider a category such as word processing software:

- Word processing software was available for minicomputers and dedicated word processing machines before personal computers even existed. Were early word processing programs for personal computers innovative?

- Was the integration of spelling checkers (and other features of modern word processors—table handling, equation editors, graphics editors, etc.) innovative?

---

[149] Some products related to Linux might be considered innovative. For example, projects to link together large clusters of relatively inexpensive Linux computers into a single "supercomputer" show promise. See footnotes 232 and 233.

[150] "BIND was originally written around 1983-1984 for use on Berkeley Software Distribution (BSD 4.3 and later releases) of UNIX by a group of graduate students at the University of California at Berkeley under a grant from the US Defense Advanced Research Projects Administration (DARPA). … The Internet Software Consortium has outsourced the development work on BIND 9 to Nominum, Inc." Nominum, "BIND (Berkeley Internet Name Domain)," 2001, http://www.nominum.com/resources/whitepapers/bind-white-paper.html (downloaded April 16, 2002).

[151] "Red Hat Sendmail HOWTO: 1.2 History," 2000, http://www.redhat.com/support/resources/howto/RH-sendmail-HOWTO/x29.html (downloaded April 25, 2002).

[152] Perl explicitly acknowledges "ancestors" like C, awk and sh. (Elaine Ashton, "The Timeline of Perl and its Culture," http://history.perl.org/PerlTimeline.html (downloaded April 2, 2002).)

[153] "Apache HTTP Server Project," http://httpd.apache.org/ABOUT_APACHE.html (downloaded April 16, 2002).

[154] "Apache HTTP Server Project," http://httpd.apache.org/ABOUT_APACHE.html (downloaded April 16, 2002).

- Was the development of graphical word processors (true "what you see is what you get") innovative?

The breadth and depth of commercial software that is currently available came, in general, from investments made in pursuit of profits. Many products, of course, have copied (and improved upon) features of their competitors—that is the nature of the competitive process. But the spreadsheets, word processors, presentation graphics, multimedia encyclopedias, video games, graphics arts, and other commercial products of today bear little or no resemblance to their forebears (if any) of 25 years ago. Clearly, much innovation in commercial software has occurred over those 25 years. Just as clearly, much (but certainly not all) of the focus of GPL software over the past two decades has been on creating "free" versions of proprietary software.

C. The Future Evolution of Open Source without Government Favoritism

Absent government favoritism for (or against) open-source software, software users will choose the software that they believe best suits their needs, taking into account price, quality, ease of use, support, and other characteristics that they consider important. For some types of customers and software, proprietary software is likely to be more successful; for other types of customers and software, open-source software is likely to be more successful. As discussed above, nothing in the history of commercial software suggests that self-interested consumers will make ill-informed decisions—software market leaders can and do get replaced.[155]

Open source has proved useful in numerous areas managed by the technically adept: operating systems, file servers, Web servers, mail servers, and development tools for all of the preceding. Open-source products have made substantially less progress in areas of interest to mass consumers, who value ease of use far more than do the technically adept in these areas, although it will certainly continue to provide competitive pressure to developers of proprietary software. This divide will likely persist—the open-source production method is geared for

[155] See footnote 56.

serving the technically adept, not for serving mass-market needs.[156] Even some major proponents of open-source software take that view.[157]

Whatever happens to open source generally, Linux appears to have taken on a life of its own. Linux may succeed at something that has eluded hardware and software vendors for over a decade: unifying Unix. For a variety of reasons, Unix fragmented into many not-quite compatible flavors in the 1980s and 1990s. Some attempts were made to unify Unix,[158] but today the integrated server vendors and integrated workstation vendors continue to ship their own flavors of Unix: Solaris, AIX, HP-UX, Irix, Tru64 Unix, and so forth. Some of these vendors, most notably IBM, are now interested in Linux. Others are less so. For example, Sun uses Linux on low-end servers based on Intel-compatible hardware, but it uses only its own version of Unix on all computers based on its own SPARC processor family.

An advantage of supporting Linux is that a computer vendor can rely on others to incur most of the costs of developing the operating system; the computer vendor then only incurs costs of fine-tuning Linux for its own hardware. A disadvantage is that relying on Linux would eliminate the operating system as a point of differentiating the vendor's integrated product from those of its competitors (one of the reasons why Unix fragmented). Although Linux may have the potential to unify Unix, it is not clear how great that potential is. Moreover, Linux faces its own potential for fragmenting, as discussed above.

---

[156] Open source has potential in another important area: "embedded devices." These include set-top boxes for advanced cable television services, cash registers, ATMs, personal digital assistants (PDAs, like the Palm family of products), and more. These devices need an operating system; many also need at least some degree of network or even Internet connectivity. Since these devices tend to have relatively fixed user interfaces, they do not face the usability requirements of operating systems on PCs. Further, developers of the hardware often need operating systems that are highly customized to their hardware; open-source operating systems such as Linux permit them to do their own customization. But many other operating systems are already widely used for embedded devices.

[157] See, e.g., Brian Behlendorf, "Open Source as a Business Strategy," 1999, http://www.oreilly.com/catalog/opensources/book/brian.html (downloaded May 20, 2002), from Chris DiBona et al (editors), *Open Sources: Voices from the Open Source Revolution,* (O'Reilly and Associates, 1999).

[158] "UNIX Past," June 25, 2001, http://www.unix-systems.org/what_is_unix/history_timeline.html (downloaded April 5, 2002).

VI. Government Interventions in the Software Market to Assist Open Source

Governments use significant amounts of computer software. The U.S. government alone spent $3.7 billion on prepackaged software in 2000.[159] State and local governments spent another $4.5 billion on prepackaged software in 2000. Like businesses, governments usually make decisions to use proprietary or open-source software for particular applications based on the merits. In some cases, open-source software is better than proprietary software—on price, technical advantages, or both grounds. In other cases, proprietary software may be the best choice because its technical superiority outweighs the fact that the open-source alternative provides the source code for free. Or there may be no open-source alternative. For example, in Germany the Bundestag recently decided, based in part on a study it commissioned, to use Linux on most servers while using Windows for clients on desktops.[160] While one could argue whether this is the right decision, and the pressure on the Bundestag to consider the issue had other motives, there is nothing that distinguishes such decisions from similar decisions made by profit-maximizing businesses.

However, proponents of open source have lobbied governments around the world to provide various preferences for open source software. Richard Stallman recently spoke about copyright and open source software before the Brazilian Congress.[161] Not all of these efforts have come from members of the open-source community. For example, Lawrence Lessig, a distinguished American law professor, has argued that governments should support open source.[162] Governments have established study groups to consider government support for open source, and some politicians in many countries have introduced legislation to help open source. Few governments, however, have to date enacted explicit preferences for open source software—a handful of cities in Brazil are the most prominent.

---

[159] "2001 Annual NIPA Revision," Bureau of Economic Analysis, and the U.S. Department of Commerce, August 2001, Tables 1, 11, http://www.bea.doc.gov/bea/papers/tables.pdf.

[160] See http://www.heise.de/newsticker/data/anw-14.03.02-012/ (German).

[161] Free Software Foundation Press Release, "Richard M. Stallman Addresses Brazilian Congress on Free Software and the Ethics of Copyright and Patents," March 20, 2001, http://www.gnu.org/press/2001-03-20-Brazil.txt (downloaded April 16, 2002).

[162] Lawrence Lessig, *The Future of Ideas*, Random House: New York (2001), p. 247. See also footnote 7.

This section examines whether there is an ***economic*** rationale for having governments provide preference to open-source software. Are there reasons to believe that market competition between proprietary and open-source software will fail to achieve the socially optimal mix of these two kinds of software? If so, are there reasons to believe the government interventions can increase social welfare by giving open-source software some form of boost? If not, to what extent could government interventions in favor of open-source reduce social welfare?

Part A explains the economic approach to analyzing whether government intervention into the marketplace is desirable. The answer turns on whether there is a market failure, and whether there is a government solution that can make things better. Part B presents a survey of government proposals and initiatives concerning open source. The public rationales for preferring open source range from the purely technical to the purely ideological. Only a few of these rationales involve even the suggestion that there is a market failure that needs to be corrected. Part C considers the two possible economic arguments for granting preferences to open source: that open source is more innovative but would otherwise be underfunded in a market economy and therefore should get special treatment; and that the government should encourage open source software—especially Linux—to provide a competitive alternative to proprietary software—generally with Microsoft in mind. Part D evaluates from the standpoint of economics a particular government policy—requiring that the results of certain government-funded software development be issued under the GPL.

A. The Economic Approach to Government Intervention

Modern economics starts with the proposition that market forces generally do a rather good job by themselves at maximizing social welfare—measured, roughly speaking, as the value that society gets from its scarce resources. There is a body of theoretical literature, starting with Adam Smith's *Wealth of Nations*, that explains how the selfish actions of individuals and businesses result in the production and allocation of goods and services in a way that tends to make the group as well off as possible. As Smith put it, every individual is "led by an invisible hand to promote an end which was no part of his intention. Nor is it always

the worse for the society that it was no part of it. By pursuing his own interest he frequently promotes that of the society more effectually than when he really intends to promote it."[163] Modern mathematical models explain how this decentralized process maximizes the collective good in formal terms.

A wide variety of practical experience generally supports the proposition—at a very gross level—that market forces, and economies in which these forces are left largely unfettered by government involvement, are better than the alternatives at maximizing social welfare. Much of the 20th century was devoted to the grand experiment of whether controlled versus capitalistic economies worked best; it ended with the large-scale collapse of the controlled ones and the boom of the capitalist ones. Within the capitalist economies, government efforts to regulate, or in some cases run, major industries such as the post office, telecoms, airlines, and energy lead to great dissatisfaction and efforts, beginning in the mid 1970s, to reduce greatly government involvement. The United States and Great Britain lead this trend, but other countries, such as France and Japan, have followed suit.

This is not to say that governments do not have any role in the economy or that interventions by the government cannot improve on market forces in some circumstances. Economists have identified two major conditions that are both necessary for intervention to make the public better off economically. The first is the identification of a market failure—some explanation as to why the market, which we presume to be most efficient most of the time, does not work. The second is the identification of a government solution that is likely to make the public better off by correcting the market failure, at reasonable cost, and without introducing other problems.

Economists have identified many theoretical situations in which market forces may not maximize public welfare. Indeed, a great deal of research in economics in the last quarter century has shown that Adam Smith's Invisible Hand is not quite as benevolent as he suggested. The Nobel Prize in Economics was awarded in 2001 to three economists—George

[163] Adam Smith, *An Inquiry Into the Nature and Causes of the Wealth of Nations*, Ed. Edwin Cannan, New York: Random House, 1937, Book 4, Chapter 2, p. 423.

Akerlof, Joseph Stiglitz, and Michael Spence—who have identified a raft of problems over the years.[164] This is not the place to go into the litany, but here are some examples.

1. Market economies tend not to produce and disseminate enough technical knowledge. On the one hand, they may not produce enough because once technical knowledge is produced, it is often technically easy for others to share in the benefits without paying for the costs; if incentives do not exist to produce technical knowledge, it will not be produced. Once technical knowledge is produced, market economies may not disseminate it enough because those who produce it may keep it a secret—even though it is costless to disseminate—to protect their investment. The patent system is an imperfect government method for remedying these failures—the government gives inventors a temporary monopoly over their inventions in exchange for full disclosure.

2. Market economies find that in some industries only one firm can survive—there is a natural monopoly in the sense that one firm can serve consumers more efficiently than two could; but undisciplined by competition, this monopoly may charge too much and produce too little relative to what best serves the public. Public utility regulation is an imperfect method for fixing this problem—historically regulators have limited profits and prices. Market methods—such as auctioning off monopolies—have become more popular over the years, and intrusive government regulation less popular.

3. Private businesses do not consider the costs they impose on society when they emit toxic substances—these "negative externalities" can be controlled by government regulation.

At least since Ronald Coase's work,[165] however, economists have recognized that government solutions do not always make things better. First, there may not be a government solution that actually fixes the market failure. Second, if there is, that solution may require the creation of a costly government bureaucracy. Third, that solution may have all sorts of side effects that cause market failures that cost the public more than the market failure the solution was designed to remedy. Fourth, the theory of rent-seeking suggests that, in certain cases, coalitions can work the political process to get government interventions that benefit the coalition at the expense of the public at large. These coalitions often justify interventions on the grounds that it is needed to remedy a market failure. So the fourth problem results from erroneously and expensively fixing a market failure that may not exist in the first place.

[164] See The Royal Swedish Academy of Sciences, "The 2001 Sveriges Riksbank (Bank of Sweden) Prize in Economic Sciences in Memory of Alfred Nobel," Press Release, October 10, 2001, available at http://www.nobel.se/economics/laureates/2001/press.html.

[165] Ronald Coase, "The Problem of Social Cost." *Journal of Law and Economics*, Vol. 3, October 1960, pp. 1-44.

That government cures may be worse than the disease is widely acknowledged by economists. For example, Professor Stiglitz has written:

> When there is a market failure, there is a *potential* role for government. Government needs to consider each of the alternatives… and assess the likelihood that one or the other alternative will succeed. Such an assessment may conclude that it is better not to intervene after all. Recent decades have provided numerous examples of government programs that have either not succeeded to the extent their sponsors had hoped, or failed altogether.[166]

Professors Michael Katz and Harvey Rosen have written that:

> It must be emphasized that while efficiency problems provide opportunities for government intervention in the economy, they do not require it. That the market-generated allocation of resources is imperfect does not mean that the government can do better. For example, in certain case the costs of setting up a governmental agency to deal with an externality could exceed the cost of the externality itself. Moreover, governments, like people, can make mistakes. Indeed, some argue that the government is inherently incapable of acting efficiently, so that while in theory in can improve upon the status quo, in practice it never will. While extreme, this argument does highlight the fact that the First Welfare Theorem is helpful only in identifying situations in which intervention may lead to greater efficiency.[167]

As with most real-world markets, software markets do not work as perfectly as a benevolent central planner would like. To begin with, most of the work in economics that demonstrates the optimality of markets is based on assumptions that do not apply to software. Most of this work applies to markets, such as that for wheat, in which thousands of producers compete with each other, with full disclosure of information on prices and quality, in a more or less static environment. The production of software involves substantial fixed costs, the creation of intellectual property, and the likelihood that prices will substantially exceed marginal costs.[168] All this takes place in a dynamic market subject to rapid rates of technological change.[169] Economists have only begun to analyze the properties of this kind of

---

[166] Joseph E. Stiglitz, *Principles of Microeconomics*, New York: W. W. Norton and Company, Inc., 1997, p. 163.

[167] Michael L. Katz and Harvey S. Rosen, *Microeconomics*, Boston: Irwin/McGraw-Hill, 1998, p. 399.

[168] "Price = marginal cost" is the condition for socially optimal production in the simple textbook model of industry.

[169] Kenneth G. Elzinga and David E. Mills, "PC Software," The Antitrust Bulletin XLIV (1999) 739-786; David S. Evans and Richard Schmalensee, "Some Economic Aspects of Antitrust Analysis in Dynamically Competitive

(continued...)

competition formally, and the work has progressed slowly because the problem is actually quite difficult.[170]

Nevertheless, the history of the software industry suggests that it has worked quite well from the standpoint of consumers. We saw earlier that output has increased rapidly, quality-adjusted prices have fallen, and innovation has been rapid. Although firms have dominated particular categories of software, at least for a time, the industry is fairly unconcentrated and is generally thought of, among its participants, as highly competitive.[171] There is a considerable amount of entry into the industry, as well as exit, and the key players turn over with some frequency. Economists take these characteristics as showing the competitive health of an industry.[172] Of course, there could be problems that need fixing and solutions worth considering.

(...continued)

Industries," in *Innovation Policy and the Economy, Volume 2*, edited by Adam B. Jaffe, Josh Lerner and Scott Stern, The MIT Press: Cambridge, MA, 2002.

[170] Economists have used models of dynamic competition to examine many different economic issues including patent races, standards and compatibility, and diffusion of technology. See, e.g., Michael Katz and Carl Shapiro, "Network Externalities, Competition, and Compatibility," 75 American Economic Review, June 1985, pp. 424-40; Michael Katz and Carl Shapiro, "R&D Rivalry with Licensing or Imitation," 77 American Economic Review, June 1987, pp. 402-20; Gene Grossman and Carl Shapiro, "Dynamic R&D Competition," 97 Economic Journal, June 1987, pages 372-87; Drew Fudenberg and Jean Tirole, "Pricing a Network Good to Deter Entry," 48 Journal of Industrial Economics, December 2000, pages 373-90; Jennifer F. Reinganum, "The Timing of Innovation: Research, Development, and Diffusion," Richard Schmalensee and Robert D. Willig, eds. *Handbook of Industrial Organization*, Volume 1, New York: Elsevier Science, 1989, pages 849-908; Jean Tirole, *The Theory of Industrial Organization*, Cambridge: The MIT Press, 1988, Chapter 10; and Paul Stoneman, *Economic Analysis of Technological Change*, Oxford University Press, 1983. However, others note that it is extremely difficult to model many aspects of competition in these types of industries. See, e.g., John Sutton, *Technology and Market Structure: Theory and History*, Cambridge: The MIT Press, 2001, Ch. 14-15.

[171] See e.g., Kenneth G. Elzinga and David E. Mills, "PC Software," 44 *The Antitrust Bulletin* 739 (1999). See also Microsoft Corp., Form 10-K for Fiscal Year Ended June 30, 2001 ("The software business is intensely competitive and subject to rapid technological change"); Oracle Corp., Form 10-K for Fiscal Year Ended May 31, 2001 ("The computer software industry is intensely competitive and rapidly evolving"); Sun Microsystems, Form 10-K for Fiscal Year Ended June 30, 2001 ("We compete in the hardware and software products and services markets. These markets are intensely competitive."); SAP AG, Form 20-F for Fiscal Year Ended December 31, 2001 ("The software and Internet industry is intensely competitive.").

[172] Dennis W. Carlton and Jeffrey M. Perloff, *Modern Industrial Organization*, 3rd Ed., Reading, MA: Addison-Wesley, 2000, pp. 56-58.

B. Governments Proposals and Initiatives Concerning Open Source

To understand current government thinking on open source, we have conducted a survey of proposals and initiatives around the world. This section presents the highlights. It begins by surveying some of the key initiatives and then summarizes the most frequently mentioned rationales for government to support these initiatives.

We begin with politico-economic observations: as users of software, governments face daily decisions about what software to use—decisions that, in general, are no different than the decisions that must be made by countless private firms and individuals around the world. When legislators get involved, however, these decisions have moved from the strictly technical/economic arena to the political. Much the same is true when administrators set up special commissions to consider whether to institute government policies that favor open source. Decisions based on the merits would not need such special commissions—private firms and individuals make their decisions without commission recommendations. As a result, the existence of special commissions, legislative proposals, and the like demonstrates an interest in having government decisions made on grounds other than those that form the basis of private decisions—regardless of whether any resulting recommendations claim to be based on technical/economic criteria.

1. Initiatives

The European Commission and the European Parliament have initiated a number of studies of open source. Many of these studies have touted the benefits of open source, and some have recommended affirmative efforts to expand the use of open source by the Commission and by the Member States. The European Commission and the European Parliament, however, have not approved any serious programs that would promote open source at the expense of proprietary software. To take one example, the Commission's Interchange of Data between Administrations (IDA) program issued a study in June, 2001, that concludes that open source software is "still not extensively used in most of the European Member States' public administrations," but "on general-purpose servers as well as on office desktop, Open Source software will present tomorrow the most realistic, and sometimes the only real technical and

economical alternative to Microsoft products."[173] The study also argues that "the requirement of the use of [open source software (OSS)] could be justified under Article 81, paragraph 3" of the EC treaty;[174] "software patents present a major threat concerning a fundamental liberty";[175] "OSS is considered to better respect standards";[176] "OSS in general and GPL in particular permits a greater rate of innovation, with greater efficiency";[177] "with OSS you will have no (or less) backdoor(s), no electronic spy that may be totally hidden somewhere in the software."[178] Thus far, the Commission has not acted on these findings, which appear to have both economic and ideological elements.

To take another example, on September 5, 2001 the European Parliament adopted a Resolution that "urges the Commission and Member States to … promote … European encryption technology and software and above all to support projects aimed at developing user-friendly open-source encryption software." Also, it "calls on the Commission and Member States to promote software projects whose source text is made public (open-source software)," and it asks "the Commission to lay down a standard for the level of security of e-mail software packages, placing those packages whose source code has not been made public in the 'least reliable' category."[179] The Parliament's recommendations are only that, and as far as we know the Commission has not done anything to follow them.

---

[173] "Study into the Use of Open Source Software in the Public Sector," Part 3, IDA, June 2001, p.7, http://europa.eu.int/ISPO/ida/jsps/index.jsp?fuseAction=home (downloaded May 20. 2002).

[174] IDA (2001) at 68. Article 81 (1) prohibits "…all agreements between undertakings, decisions by associations of undertakings and concerted practices which may affect trade between Member States and which have as their object or effect the prevention, restriction or distortion of competition within the common market…" Paragraph 3 exempts from the scrutiny of paragraph 1 those practices that contribute to "…improving the production or distribution of goods or to promoting technical or economic progress, while allowing consumers a fair share of the resulting benefit …" IDA argues that the use of OSS could be justified as the practice that promotes "technical or economic progress, allowing consumers a share of the resulting benefits."

[175] IDA (2001) at 74.

[176] IDA (2001) at 15.

[177] IDA (2001) at 39.

[178] IDA (2001) at 16.

[179] The European Parliament Resolution on the Existence of a Global System for the Interception of Private and Commercial Communications, September 5, 2001, http://www3.europarl.eu.int/omk/omnsapir.so/pv2?PRG=CALEND&APP=PV2&LANGUE=EN&TPV=PROV&FILE=010905 (downloaded May 20, 2002); see also "European Parliament Adopts 'Echelon' Report," available at http://www.cnn.com/2001/TECH/internet/09/07/echelon.report.idg/ (downloaded May 20, 2002).

The German government, however, has undertaken concrete initiatives to promote open source and, along with France, is one of the most active national governments in this field. On March 14, 2002, the Council of Elders, a joint deliberative body whose task is to manage the internal affairs of the Bundestag, arrived at a decision on the new Bundestag IT environment. It decided to follow the luK (11-member committee comprised of representatives from major political parties) recommendation that proposed the installation of Linux on approximately 150 servers and Windows XP on 5,000 desktops.[180] The luK recommendation was based on the INFORA study commissioned by the Ministry of Interior, which analyzed various scenarios of Bundestag IT environment.[181] As noted earlier, the decision does not necessarily amount to a government intervention—governments, just like businesses, must make information-technology decisions. However, numerous political statements in favor of open source accompanied the debate in the Bundestag over the migration issue.[182] These statements demonstrate that the bias towards open source is not based entirely on technical and economic considerations.

The Bundestag had earlier passed a resolution on "Germany's Economy in the Information Society" on November 9, 2001 explicitly to promote open source software in the federal administration. Supported by the Social Democrats and Greens, the resolution describes open source as a means to secure competition against dominant players in software markets. It also lists claimed advantages of open source: stability, better potential to be tailored to users' needs, and high security. The resolution calls on the government to introduce open source in the federal administration and states that open source should be used wherever this will lead to

[180] See http://www.heise.de/newsticker/data/anw-14.03.02-012/ (downloaded May 20, 2002) (German).

[181] See http://www.heute.t-online.de/ZDFheute/artikel/0,1251,COMP-0-178460,00.html (downloaded May 20, 2002) (German); see also http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/jk-07.12.01-008/default.shtml&words=Open%20Source (downloaded May 20, 2002) (German).

[182] For instance, a Social Democrat, Jörg Tauss, announced, "my wish would be to declare the entire Bundestag a Microsoft-free zone." See http://www.my-opensource.org/lists/myoss/2002-03/msg00010.html (downloaded May 20, 2002). The Greens stated that open source represents a special chance for the European software segment, since for the first time the United States is not leading in the field. They also stated, "*Open Source entspricht der grünen Philosophie von Transparenz, Bürgerbeteiligung und Partizipation*." ("Open Source corresponds to the green philosophy of transparency and citizen participation.") See http://www.gruene-fraktion.de/rsvgn/rs_dok/0,,669,00.htm (downloaded May 20, 2002) (German). Steffi Lemke, the representative of the Greens, announced that the decision to migrate towards Linux in the Bundestag was the first important step towards the adoption of open source. See http://webnews.html.it/focus/181.htm (downloaded May 20, 2002).

cost savings. Again, intervention by legislators into an administrative issue reflects an attempt to interject political considerations into what should be a technical/economic decision. The resolution considers open source as a special opportunity for the European software industry.

Since the beginning of 1999, the French public sector has widely adopted open source solutions and continues to move in the direction of complete open-source infrastructure.[183] Several governmental institutions have already switched to open-source software. For example, the Ministry of Culture and Communication has started a massive migration towards Linux. Its stated objective is to achieve full open-source infrastructure by 2005.[184] Other agencies moving to open source include the Ministry of Justice, the Department of National Education, and the Ministry of Economy, Finance and Industry.[185]

Although these may well have been decisions on the merits, a number of proposals have been made in France that open source should be chosen because it helps achieve other social objectives. Two bills, neither of which was enacted into law, illustrate the views on open source. One parliamentary bill submitted in December 1999 would require that, after a short transition, all software used by the government should be open code and would also create an open source agency to oversee the transition process.[186] Another bill, submitted in May 2000, made a number of points about software: 1) software should not come from only one maker and therefore must be open code so as to be compatible with software from other makers; 2) future use of software should not depend on the good will of the manufacturer and therefore the source code should be available; 3) open code software would permit users to detect attempts to "spy" on them; and 4) principles of compatibility and competition in software and respect for privacy and civil liberties should be established.[187]

---

[183] IDA (2001), at 28.

[184] See http://www.linux-mandrake.com/fr/pr-culture.php3 (downloaded May 20, 2002) (French).

[185] IDA (2001), at 33-35.

[186] Parliamentary Bill, available at http://www.senat.fr/leg/ppl99-117.html (downloaded May 20, 2002) (French).

[187] Parliamentary Bill, available at http://www.assemblee-nat.fr/propositions/pion2437.asp (downloaded May 20, 2002) (French).

Efforts to require governments to use open source have succeeded in a few locales in Brazil.[188] In June 2001, the City Council of Amparo, a city in the state of São Paulo, passed a law requiring that the municipal government prefer free, unrestricted open source software.[189] The municipality of Recife also has an open source preference pursuant to an executive decree issued by the mayor in April 2000.[190] In December 2001, the Chamber of Councilmen in Porto Alegre approved a project that sets the conditions for the use of open source in the municipal administration.[191] Several other Brazilian cities and states have considered or are considering open source preference proposals.[192] In Italy and Spain, resolutions favoring the use of open source software have also been passed. In Italy, the city government of Florence passed a resolution warning that the use of proprietary software was leading to "the computer science subjection of the Italian state to Microsoft."[193] In May 2002, the Council of Pescara approved a motion, introduced by the Italian Communist Party and the Left Democrats, asking for introduction and development of Open Source Software in public administration of the Province of Pescara.[194] In Spain, the Canary Islands Regional Parliament approved a proposal that would urge the regional government, in partnership with local authorities and companies, to promote use of OSS through training courses and increasing of public awareness about OSS availability.[195]

---

[188] *See* Paul Festa, "Government's push open-source software," *CNet News.com*, August 29, 2001, http://news.com.com/2100-1001-272299.html?legacy=cnet (downloaded May 20, 2002).

[189] *See* http://www.linuxon.com.br/noticias/270601.asp (downloaded May 20, 2002) (Portuguese); see also http://www.at.com.br/2206/politica.htm (downloaded May 20, 2002) (Portuguese).

[190] *See* http://www.pernambuco.com/tecnologia/arquivo/softlivre1.html (downloaded May 20, 2002) (Portuguese).

[191] *See* http://www.softwarelivre.rs.gov.br/index.php?menu=maisnoticias&codigonoticia=1008350031 (downloaded May 20, 2002) (Portuguese).

[192] The cities include Curitiba, Florianópolis, São Paulo and the states include Rio de Janeiro, Bahia, Espirito Santo, Minas Gerais, Parana, Pernambuco, São Paulo, and Rio Grande do Sul. *See* http://www.softwarelivre.rs.gov.br/index.php?menu=maisnoticias&codigonoticia=1008350031 (downloaded May 20, 2002) (Portuguese); see also http://news.com.com/2100-1001-272299.html (downloaded May 20, 2002).

[193] Paul Festa, "Government's push open-source software," *CNet News.com*, August 29, 2001, http://news.com.com/2100-1001-272299.html?legacy=cnet (downloaded May 20, 2002).

[194] See http://www.interlex.it/pa/ppescara.htm (downloaded May 20, 2002) (Italian).

[195] See http://www.parcan.es/pub/Bop/5L/2001/166/bo166.pdf (downloaded May 20, 2002) (Spanish).

Elsewhere, proposals on open source software use are in various stages. In Singapore, a government agency responsible for planning strategies to aid the economy has reportedly decided to promote Linux. The agency targets software developers, distributors, and service providers and offers economic incentives such as tax breaks and grants for Linux-related economic development. In Peru, a bill that would require the use of free software in government computers is currently in the legislature.[196] And similar bills have been proposed elsewhere in Latin America in places like Argentina.[197]

2. Rationales Offered

We reviewed the various studies and proposals and synthesized the rationales that have been presented for using or promoting open source software. Some of the rationales have involved claims that open source provides certain technical or cost-saving advantages over proprietary software. Basing purchasing decisions on such advantages does not amount to a government intervention—obviously the government should get the best software just as it should get the best tanks and the best paperclips. Other rationales seem to be based on a desire to correct a perceived market failure although the debates are seldom couched in those terms. Finally, some of the rationales are based on ideological views. Our summary is based mainly on Germany since the debate concerning open source seems to have progressed farther there than in any other jurisdiction. However, the general arguments discussed below have been made in some form by various people in many other countries.

a. Security, Stability, and Privacy

Many of the government efforts to promote open source are based on the claim that open source software is more secure or more stable than proprietary software. For example, the

[196] Julia Scheeres, "Peru Discovers Machu Penguin," Wired, April 22, 2002, http://www.wired.com/news/business/0,1367,51902,00.html (downloaded May 20, 2002); Thomas C. Greene, "MS in Peruvian Open-Source Nightmare," *The Register*, May 5, 2002, http://www.theregister.co.uk/content/4/25157.html (downloaded May 20, 2002).

[197] In September 2000, a member of the Chamber of Deputies proposed legislation that would create a procurement preference for OSS. The proposal would require that the federal government, autonomous federal agencies, and state-owned enterprises use and acquire only OSS. The bill expired in March 2002, and a new Bill of Free Software was submitted to the Chamber of Deputies of National Congress on March 27, 2002. See http://www.grulic.org.ar/proposicion/proyecto/ley-dragan/doc-asesores-3.html (downloaded May 20, 2002); http://www.grulic.org.ar/proposicion/proyecto/ley-dragan/index.html (downloaded May 20, 2002).

German government claims that Linux is one of the most stable and secure operating systems since it allows developers to examine the source code, check for problems, and correct problems more quickly.[198] A meeting on "Open Source Software in Public Administration" presented the view that knowledge of source code is a fundamental prerequisite for the protection of systems and networks, for example against viruses.[199] In its resolution on "Germany's Economy in the Information Society," the Bundestag stated that open source software is characterized by stability and high security.[200] A related concern is that "some versions of Windows contain backdoors designed to grant the U.S. National Security Agency access to users' data."[201]

b. Cost Savings

Some government-sponsored studies have claimed that using open source saves money. German experts said that the use of open source software in public administration would save €130 million for the federal government, and €2.6 billion countrywide.[202] In July 2001, the State Secretary in the Federal Ministry of Economics and Technology said that Linux was "more stable, cheaper, more customisable and more secure" compared to proprietary software.[203]

c. Independence

On various occasions the German government has expressed its concerns about the administration's dependency on single software providers. The State Secretary in the Federal Ministry of Interior Affairs stated in July 2001 that dependence on a single software provider makes systems more vulnerable, and that the federal government would try to reduce its

[198] See http://linux.kbst.bund.de/auftakt/vortraege/zypries/ (downloaded May 20, 2002) (German); see also http://www.cnn.com/2001/TECH/industry/11/04/linux.os.idg/index.html (downloaded May 20, 2002).

[199] See http://linux.kbst.bund.de/auftakt/vortraege/zypries/ (downloaded May 20, 2002) (German).

[200] See http://linux.kbst.bund.de/bundestag/bt-pp14.199.html (downloaded May 20, 2002) (German).

[201] See http://www.cnn.com/2001/TECH/industry/10/16/german.parliament.idg/ (downloaded May 20, 2002).

[202] See http://morgenpost.berlin1.de/archiv2001/011007/fernsehen/story466199.html (downloaded May 20, 2002) (German).

[203] See http://www.heise.de/newsticker/data/odi-05.07.01-000/ (downloaded May 20, 2002) (German).

dependence on single software providers by adopting open source.[204] At the regional level, Mrs. Harms, a Green Party Member of the Lower Saxony regional Parliament, in her parliamentary question directed to the regional government, stated that a Linux platform complemented with open source and commercial products should relieve the dependence on a single provider.[205]

d. Innovation

A number of the proposals to assist open source have claimed the open source is more innovative that proprietary software. The European Commission's IDA study stated that "OSS in general and GPL in particular permits a greater rate of innovation, with greater efficiency."[206] For example, in the Spirit Project, partially funded by the European Commission's Fifth Framework Programme and established to "accelerate the uptake of open source solutions in European health care," it was mentioned that "[t]he open source approach plays a key role in accelerating the evolution and uptake of best practice solutions in health care. It also stimulates innovation and evidence-based review."[207] On a national level, in its study on patent protection of software products commissioned by the German Federal Department for Economy and Technology and published in November 2001, the Fraunhofer Institute and Max-Planck-Institute stated that "[t]he further development of Open Source as a kind of public good, that on principle is available for use by all economic units and thus in the sense of the new growth theory promotes the general technical progress and therefore innovation dynamics, is perceived to be in special danger."[208] Professor Lessig, whose views we discuss both above and below, has also based his support on the proposition that open source software has been innovative.[209]

---

[204] See http://www.bmi.bund.de/top/dokumente/Rede/ix_47733.htm (downloaded May 20, 2002) (German).

[205] See http://www.landtag-niedersachsen.de/Drucksachen/1501-2000/14-1942.pdf (downloaded May 20, 2002) (German).

[206] IDA (2001).

[207] "SPIRIT – Accelerating the Uptake of Open Source in Healthcare," by Bud P. Bruegger, et al., http://www.sistema.it/LinuxAfrica2001/ (downloaded May 20, 2002).

[208] "Micro- and Macroeconomic Implications of the Patentability of Software Innovations. Intellectual Property Rights in Information Technologies between Competition and Innovation," by Fraunhofer Institute and Max-Planck-Institute, November 15, 2002, http://www.sicherheit-im-internet.de/download/softwarepatentstudie_e.pdf (downloaded May 20, 2002).

[209] See footnotes 144 and 145.

e. Competition

The German government has argued that open source plays an important role in stimulating competition. Margaret Wolf, State Secretary in the Federal Ministry of Economics and Technology, in her speech on July 5, 2001 noted that open-source software played an important role in bringing competition to the software market.[210] Along the same line, the audit study presented to the Budget Committee of the regional parliament of Schleswig Holstein stated that Linux and Linux compatible applications should bring more competition into the IT arena.[211] The proposition that open source would help competition was also noted by the Commission's 2001 IDA study, which argued that governments could require the use of open source without running afoul of European competition laws because doing so promotes "technical or economic progress, allowing consumers a share of the resulting benefits."[212]

f. Helping Domestic Industries and Other Nationalistic Motives

The German government national interest has yet another argument in favor of open source. As Prof. Lutterbeck states in his report commissioned by the Federal Ministry of Economics and Technology, "new and not yet discussed in Germany are the figures of the worldwide occurrence of open-source developers. They show that German developers form the second largest group. On the whole, European developers are predominant." He concludes that the open-source area not yet dominated by the United States is of great economic importance for Germany.[213] In its resolution on "Germany's Economy in the Information Society" on November 9, 2001, the German Bundestag also stated that open source software should be

---

[210] See http://www.heise.de/newsticker/data/odi-05.07.01-000/ (downloaded May 20, 2002) (German).

[211] See http://www.lrh.schleswig-holstein.de/Veroeffentlichungen/IT-Landtag/it_lt.pdf (downloaded May 20, 2002) (German); http://www.lvn.ltsh.de/infothek/wahl15/aussch/finanz/niederschrift/2001/15-033_01-01.html (downloaded May 20, 2002) (German).

[212] "Study into the Use of Open Source Software in the Public Sector," Part 3, IDA, June 2001, at 68, http://europa.eu.int/ISPO/ida/?http&&&ag.idaprog.org/Indis35prod/doc/333 (downloaded May 20, 2002).

[213] "Security in Information Technology and Patent Protection for Software Products: a Contradiction?" by Prof. Bernd Lutterbeck, December 2000, http://www.sicherheit-im-internet.de/download/BMWi_Gutachten_englisch.pdf (downloaded May 20, 2002).

considered a special opportunity for the European software industry and should not be missed.[214]

g. Ideological

As noted above, most recent open-source software seems to be developed under the GPL, due to the influence of the FSF. Also as noted above, the motivation of the FSF in support of the GPL is purely ideological. Some of the justifications advanced for government policies to favor open source echo these sentiments.

For example, in his recent address to the Brazilian Congress, Richard Stallman stated: "I find in Brazil considerable awareness that free software is a social and political issue as well as a practical and economic one. The programmers and users that I have met here are very receptive to the ideas of freedom that free software represents."[215]

As one of the reasons for supporting open source software, the European Working Group on Libre Software (created at the initiative of the Information Society Directorate General of the European Commission) stated that it "provides a new forum for democratic action."[216]

C. Economic Arguments for Helping Open Source

Let us now consider some of the economic arguments for helping open source. These arguments are based on the premise that open source is "better" in some dimension and therefore the government should help promote open source. In theory one can distinguish between two sorts of arguments. One is that open source is superior and therefore should be used more by prudent purchasers—government is no different from business. The other is that open source could provide various economic benefits if it were successful, so government should give it a boost.

In practice these arguments tend to blur. Professor Lessig, for example, writes:

---

[214] See http://linux.kbst.bund.de/bundestag/bt-pp14.199.html (downloaded May 20, 2002) (German).

[215] Free Software Foundation Press Release, "Richard M. Stallman Addresses Brazilian Congress on Free Software and the Ethics of Copyright and Patents," March 20, 2001, http://www.gnu.org/press/2001-03-20-Brazil.txt (downloaded April 16, 2002).

[216] Working Group on Libre Software, "Free Software / Open Source: Information Society Opportunities for Europe?," April 2000, http://eu.conecta.it/paper.pdf (downloaded April 25, 2002).

> What reason does the government have for supporting closed code, when open code is as powerful and the externalities from using open code would benefit others? If the PCs that the government owned ran something other than Windows, then the market for these alternative platforms would be wildly expanded. And if the market for alternatives were strong, then the benefits from building for these alternatives would be strong as well.[217]

His argument therefore begins with the premise that open source is as good or better than proprietary software and ends with the conclusion that the government should promote open source. Here and in his other writings, Professor Lessig seems to be doing more than advising the government on how to run its information technology department.[218] He seems to be suggesting that the government should use open source software in ways that one would never suggest for a profit-maximizing business—or perhaps even Stanford Law School. Professor Lessig's passage, though, may be based on a market failure concept to which we return below.

The arguments for promoting open source are often similar admixtures of claims about the superiority of open source software and assertions about the munificent effects of government help. We begin with the premises and then turn to the claims that governments need to help.

1. Claims about the Superiority of Open Source Software

As we saw above, many of the proposals that the government use open-source software are based on claims that it is better than proprietary software.

a. Innovation

As we discussed above, there is no basis for claiming that the open source software has generally been more "innovative" than proprietary software. Most of the proposals for assisting open-source software have been aimed at Linux and related software for client computers,

---

[217] Lawrence Lessig, *The Future of Ideas*, Random House: New York (2001), p. 247.

[218] See, for example, Lawrence Lessig, "Code and the Commons," February 9, 1999, http://cyberlaw.stanford.edu/lessig/content/articles/works/fordham.pdf (downloaded April 24, 2002) and Lawrence Lessig, "May the Source Be with You," *WIRED*, December 2001, http://www.wired.com/wired/archive/9.12/lessig_pr.html (downloaded April 24, 2002).

which generally has been released under the GPL.[219] Much of the software released under the GPL has been, intentionally, imitative of proprietary software. Many of the GPL projects underway involve further efforts to copy proprietary software. The open-source products with the strongest claims to being innovative have been released under BSD-style licenses.

b. Security and Privacy Concerns

Claims by open-source proponents that open-source software is inherently more secure than proprietary software have at least a veneer of plausibility: the more eyeballs that are looking for security problems, the higher the probability that problems will be identified and solved. We take no position on whether this argument is correct. We do note, however, that not everyone agrees with this view. There may in fact be no particular reason to believe that more eyeballs actually are looking for security problems with open source.[220] And some commentators have asserted that widespread attacks on Windows computers connected to the Internet are motivated as much or more by the ubiquity of these computers as by their security problems.[221]

Claims by open-source proponents that open-source software is more protective of privacy than is proprietary software also have some theoretical plausibility. In theory, proprietary software has the potential to watch a user, then transmit usage data back to the software vendor for marketing or other purposes. In practice, open source cannot do this easily, since a skilled programmer who noticed such behavior could excise the code from an open-source program. Similarly, encryption software could theoretically have a "backdoor" that would let those in the know decrypt information that was supposed to be secure. Such claims are difficult to evaluate. It is certainly true that open source can generally avoid potential

---

[219] See, e.g., an article by Richard Stallman on UNESCO website http://www.unesco.org/webworld/portal_freesoft/stallman_011001.shtml (downloaded May 20, 2002). The article addresses UNESCO's support of Free Software Foundation that promotes GNU/Linux system.

[220] Robert Lemos, "Too Much Trust in Open Source?," March 20, 2002, http://zdnet.com.com/2100-1104-864256.html (downloaded April 5, 2002); Wayne Rash, "Proprietary Apps Have Security Problems—but So Does Open Source," April 4, 2002, http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2857736,00.html (downloaded April 5, 2002).

[221] See, e.g., Mitch Wagner, "Virus Attacks Linux, Windows Systems," *InternetWeek*, March 30, 2001, http://www.internetweek.com/shared/printableArticle?doc_id=INW20010330S0007 (downloaded April 24, 2002).

privacy and security problems like these. There is little evidence that proprietary programs actually engage in such activity.[222] Much of the argument in this area seems based on speculation (what software theoretically could do) rather than fact (what software actually does).[223] Proprietary software producers, it should be noted, have strong financial incentives not to intrude on the privacy or compromise the security of their licensees. If such behavior became known, people would seek other alternatives.

c. Cost Savings

We note that profit-oriented firms in the private sector tend to use proprietary software extensively in some situations (particularly for client computers), with open-source software being relatively more popular on servers. This strongly suggests that the costs savings (at least acquisition costs, if not training and support costs) of open source on the client side generally have not been adequate to overcome technical advantages of proprietary software. Where formal studies have been conducted, the results have been inconclusive. For example, the audit study presented to the Budget Committee of the regional parliament of Schleswig Holstein in Germany stated that adoption of OSS where no license costs were paid might lead to substantial cost savings. However, other costs such as training, setup, support and development should be taken into account. The study concluded that there was no clear economic winner.[224]

2. Arguments for Promoting Open Source

The less sophisticated argument for promoting open source is that it is "good" so let us have more of it. The more sophisticated argument is that government assistance for open source

---

[222] Ad-supported programs are an exception: users generally should assume that they engage in exactly this type of activity. Reports of "spyware" or "sneakware" in file sharing programs such as Kazaa have also surfaced recently. See, e.g., John Borland and Rachel Konrad, "PC Invaders," *CNet News.com*, April 18, 2002, http://news.com.com/2009-1023-885144.html (downloaded May 20, 2002).

[223] For example, assorted parties long suspected that the National Security Agency in the United States had put such a backdoor into the DES encryption method before authorizing its use in the 1970s. There is no evidence, however, that such a backdoor ever existed. *See* Steven Levy, *Crypto: How the Code Rebels Beat the Government, Saving Privacy in the Digital Age,* New York: Penguin Books, 2001, especially pp. 38-39, 56, 60-64.

[224] See http://www.lrh.schleswig-holstein.de/Veroeffentlichungen/IT-Landtag/it_lt.pdf (downloaded May 20, 2002) (German); http://www.lvn.ltsh.de/infothek/wahl15/aussch/finanz/niederschrift/2001/15-033_01-01.html (downloaded May 20, 2002) (German).

will increase competition for proprietary software (particularly Windows) and thereby benefit society through externalities. We consider each in turn.

a. Promoting Open Source because It Is "Better" than Proprietary Software

Some of the arguments for having the government promote the use of open source software seem to be based on nothing more than the observation that open-source software has certain features that are claimed superior to proprietary software. For example, in November 2001, Michel Sapin, the Minister of Public Services in France, stated, "*Les deux exigences de la deuxième étape de l'administration électronique sont donc l'interopérabilité et la transparence. Ce sont justement les deux points forts des logiciels libres.*" ("Next generation e-Government has two requirements: interoperability and transparency. These are the two strengths of open source software.")[225] Mrs. Zypries, a state secretary in Germany, stated during a meeting on "Open Source Software in Public Administration" that the knowledge of code is a fundamental prerequisite for the protection of systems and networks.[226] To the extent that open-source software is better than the proprietary alternatives for meeting government information-technology needs, one could hardly object that the government should use open source. But the amount of interest in touting the advantages of open source software and the attempt, occasionally, to legislate the use of open source implies that more is going on here than a simple technical debate.

We have several observations. First, there is no basis for claiming that open source is generally superior to proprietary software. We saw earlier that both approaches have advantages and disadvantages. One would need to evaluate open source software and proprietary software on a case-by-case, product-by-product basis. Second, there is no basis for claiming that a particular advantage of a certain kind of open-source software over the comparable proprietary software will persist. As we saw above, there is nothing intrinsic about open-source software that ensures that it will be more secure than proprietary software. Third,

[225] See http://www.fonction-publique.gouv.fr/leministre/lesdiscours/discours-200111151520.htm (downloaded May 20, 2002) (French).

[226] See http://linux.kbst.bund.de/auftakt/vortraege/zypries/ (downloaded May 20, 2002) (German).

there seems to be a suggestion in some of the arguments for promoting open source that because there is something "good" about open source—security, the fact that it is free and so forth—that the government should do something to promote this "good." Such reasoning is faulty. The market will veer toward open-source software solutions if they are superior, so there is no reason why the government needs to push the market in that direction. As we have noted earlier, governments do not have good track records at picking technology winners and losers.

For other reasons, the potential for "innovation" by open-source software is not by itself a sensible economic reason for making purchase decisions to favor open source. A private firm deciding what mix of software will best meet its needs over a relevant time horizon would not care whether some production method tends to be more "innovative" than other. Rather, it would care about specific products and their technologies.

An argument could perhaps be advanced that the open-source software method is more "innovative" than the proprietary software approach, but that developers have insufficient incentives to develop enough open-source software. We discuss this below in a section on R&D support for open source. Procurement preferences are unlikely to be a useful policy, for reasons discussed in the next section.

b. Promoting Open Source to Increase Competition

As noted above, some people have argued that governments should use Linux as their operating system for clients or servers because that would provide competition to Microsoft. In some cases this argument is based on just giving the government another source of competitive supply (much like second-sourcing provides for any business). In other cases, the argument is based on the government's helping to create a competitive alternative to Microsoft that would benefit society generally. We believe that this argument, although ultimately fallacious, is one of the most promising for having the government assist open source. Let us develop the argument a bit before we discuss the problems.

Many businesses make sure they have at least two suppliers. That provides several advantages. First, it ensures that they have a source of supply if one vendor cannot perform for some reason. For example, a business that manufactures a product that requires a particular chemical compound as a critical input may not want to find itself with a single supplier of that

input. An explanation like this can hardly apply to a software supplier, however; capacity and manufacturing problems are not relevant for software.

Second, it ensures that they will have competition. Even if one supplier is better than the other, it may be good to keep the less efficient supplier around because that provides some price discipline on the more efficient supplier. Indeed, it is theoretically possible for the benefits of having a "second source" to be so great that it is worth it for a company to buy some fraction of its requirements (but not all) from the second source even if it is getting a more expensive or lower quality product. This argument makes sense only when the company would not have access to the second supplier if it did not second source, which for software could happen only if the second supplier could not remain in business without the second source contract. If the second supplier would be available anyway, then the business can always just take bids and select the best supplier. By the same reasoning, one might argue that it makes sense for governments to use Linux operating systems even when they are not the most cost-effective choices on more narrow economic grounds. Moreover, one might argue that government procurement of open-source software could help ensure the survival of long-run competitive alternatives to proprietary software in general and Windows in particular.

There are several problems with this argument. The most fundamental problem is that procurement policies seem unlikely to be useful in supporting the development of open source, due to the non-market orientation of its development. When the source code for a product is freely available, there can be no entry barriers of economic importance into the business of "supplying" that product to potential users. In such a situation, "buying" an open-source product (or support for the product) pays for the distribution costs (and the support costs) but nothing more—and certainly not development costs for open source. This problem arises regardless of the motivation underlying the procurement preference—encouraging "innovation," encouraging a competing supplier, or whatever.

Another problem is more specific to competition between Linux and Windows. One must distinguish between the use of these products on servers and on clients. In the case of servers, Microsoft is not the dominant supplier of operating systems. Based on shipments

(hardware plus software revenues), Microsoft's share of servers was 23 percent in 2000.[227] Microsoft's share has increased over the years as the price-performance characteristics of its server operating system has improved. This has provided price and innovation pressure on the other server software competitors, such as Novell, IBM and Sun. Linux has also done quite well in gaining server installations—its share of server shipments stood at 3 percent in 2000, up from 1 percent in the previous year.[228] We do not see any basis for having the government choose Linux other than on the merits in this kind of market setting.

Microsoft does, however, have a very large share of client operating systems. On single-user computers, its share of new shipments was 93 percent in 2000.[229] Some might argue that it makes sense to try to develop a competitive alternative. There are several problems with having governments do that:

1. Short of most governments agreeing collectively to support Linux, no single government—even the U.S. government—is a large enough purchaser of client operating systems to have much effect on the success of Linux; moreover, as discussed above, procurement policies seem particularly ill-suited for promoting the development of open source. Germany, much less the City of Florence, could not have much effect on the development of Linux.

2. The open-source software method does not appear likely to serve consumer needs on the client side. Unlike server operating system, which are often maintained by technically adept individuals, client operating systems are generally used by technically unsophisticated individuals. As we discussed earlier, the open-source production method does not have any mechanism for, or provide any incentives for, designing software for the masses.

3. There is no apparent reason why the "second source" alternative, if there is to be one, should be Linux. The MacOS or OS/2 could be better potential choices. They already have applications written for them, and they are produced by proprietary companies that have the incentives, and knowledge, to produce consumer-oriented software.

4. There are benefits to standardizing on client-side software. As mentioned above, many types of software (including client operating systems and some categories of client applications) exhibit "network effects." Users gain when their learning about user

[227] Data from IDC Server Tracker database.

[228] Data from IDC Server Tracker database.

[229] IDC Report #25118, "Worldwide Client and Server Operating Environments Market Forecast and Analysis Summary, 2001–2005," August 2001, Table 2, p. 12.

> interfaces, program capabilities, and the like can be readily transferred from computer to computer, home to work, and job to job. Deliberately avoiding widely-used software reduces these consumer benefits. If a large government customer shifts its use of operating systems from one to another, it will increase the network effects (to the general public) for its new operating system—but it will necessarily reduce the network effects for remaining users of its former operating system. If the new operating system started with a smaller network than the old, then the total network effects are likely to decline with such a switch (unless the new operating system actually is superior to the former one—in which case there is unlikely to be any need for government policy, as the market is likely to switch on its own).

Let us now return to Professor Lessig's suggestion that the government use open source.[230] Filling in his thoughts, the argument seems to run along the lines that: (a) if the government promotes the Linux operating system; (b) that would increase the share of computers running the Linux operating system; (c) more applications would be written to run on the Linux operating system; (d) there would be more competition for Windows; and (e) consumers would be better off. One could make similar arguments for many products with network effects that the government uses—from telecommunications systems to credit cards. Professor Lessig provides neither theory nor fact in support of having the government aid this particular product—Linux (or any other open-source software product)—and our discussion above shows why such government efforts would be neither prudent nor successful.

D. Releasing Software R&D Under the GPL

Some governments, such as in the United States, have long provided support for software R&D, such as through universities and government research laboratories. In general, government sponsorship of R&D efforts, for other industries as well as software, is based on notions of market failure and externalities:

- Research into some "desirable" fields will have spillover effects that benefit parties other than the ones doing the research.
- As a result, the parties doing the research cannot capture all of the social benefits of the research.

[230] Lawrence Lessig, *The Future of Ideas*, Random House: New York, 2001, p. 247.

- From a social cost-benefit perspective, there may therefore be reason to believe that, without government support, R&D will fall short of socially desirable levels.

- If suitable government support programs can be designed, society may be better off with some government sponsorship of R&D.

In this article, we do not address the issue of whether such R&D support for software is good public policy. We do, however, address the issue of whether R&D support for GPL software is good public policy.

In the past, U.S.-sponsored research into software either went into the public domain, remained in the hands of the military, or was spun off for commercial purposes.[231] We make no claims about specific products, but we note that the Internet arose from U.S.-sponsored research into hardware and software running and using packet-switched networks. Whatever policies were in place at the birth of the Internet, they seem to have succeeded technologically.

In recent years, however, substantial government R&D support has been directed at GPL software. For example, the Beowulf clustering software for Linux, released under the GPL, was originally developed by NASA.[232] More advanced clustering software was developed at Sandia National Laboratories, also released under the GPL.[233] The next version of the Reiser File System is sponsored primarily by the Defense Advanced Research Projects

[231] For example, throughout the 1960s the Advanced Research Projects Agency (ARPA), which was part of the Department of Defense, conducted research on communications. In 1969, it created a network of computers called ARPANet, which was designed to allow continued communications in the event of a nuclear attack. ARPANet and related research formed the basis for today's Internet. The National Science Foundation (NSF), which maintained the main Internet backbone and subsidized domain names, first allowed use of NSFNet for commercial purposes in 1991 and fully shifted its responsibilities to the private sector in 1995. See http://www.nsf.gov/od/lpa/nsf50/nsfoutreach/htm/n50_z2/pages_z3/28_pg.htm#answer3 (downloaded May 20, 2002).

[232] Thomas Sterling, "Beowulf Linux Clusters," http://beowulf.gsfc.nasa.gov/tron1.html (downloaded April 5, 2002).

[233] "Laboratories Support, Facilities, and Human Resources," February 28, 2002, http://www.sandia.gov/LabNews/LN02-22-02/LA2002/la02/support_story.htm (downloaded April 24, 2002).

Agency (DARPA) and will be licensed under the GPL.[234] And, in April 2002, Sandia National Laboratories released its "DAKOTA Toolkit" under the GPL.[235]

There seems to be no economic justification for this support of the GPL. In other areas, universities and government research laboratories have been encouraged over the past 20 years to spin off research into commercial products, particularly through the licensing of patents that emerge from their research.[236] If such policies are appropriate for other fields, there is no reason to believe that they are inappropriate for software. The justification for such policies is essentially that firms with intellectual property rights will have profit incentives to develop products and services that others will value. Support of GPL projects is incompatible with commercial spin-off efforts, since the GPL is incompatible with proprietary, commercial software.[237]

If for some reason the standard commercialization approaches appropriate for other fields of research are inappropriate for software R&D (and we know of no such reasons), then licenses less restrictive than the GPL, such as the BSD license or even the public domain, would seem appropriate: they let anyone use the software in any ways they choose; whereas the GPL sharply restricts the ways in which the software can be used. As noted above, the GPL is sometimes claimed to be more appropriate than other license types for a software "cooperative," because it might possibly encourage more "sharing" behavior among the cooperative members. But that argument cannot apply to government-sponsored R&D, which is getting funding from the government and is not part of a user "cooperative."

---

[234] Hans Reiser, "ReiserFS v.3 Whitepaper," http://www.reiserfs.org/content_table.html (downloaded April 8, 2002).

[235] Sandia National Laboratories Press Release, "Sandia's DAKOTA Toolkit on Web and Available for Free," April 8, 2002, http://www.sandia.gov/media/NewsRel/NR2002/DAKOTA.htm (downloaded April 16, 2002).

[236] The Patent and Trademark Law Amendments Act (also known as the Bayh-Dole Act) of 1980 encouraged universities and small businesses to commercialize inventions by permitting exclusive licensing of intellectual property that was developed with public funding. In exchange for the right to elect title to an invention, the licensor must agree to properly manage the invention and provide reports to the government. Since the passage of Bayh-Dole, universities have increasingly set up technology transfer programs and actively patented and commercialized inventions. See Council of Government Relations, "The Bayh-Dole Act: A Guide to the Law and Implementing Regulations," September 1999, http://www.cogr.edu/bayh-dole.htm (downloaded May 20, 2002).

[237] And, indeed, the GPL is incompatible with any assertion of patent rights.

We argue here that it is bad public policy for the government to support software R&D that is licensed under the GPL. In an extended endnote, Lessig appears to disagree with this position; he presents what he claims are arguments advanced by others in support of our position (although not the arguments we present here), and then rejects them.[238] He characterizes these arguments by others as: "Government funds should not promote a coding project that is not wholly free for anyone to take and do with as they wish." He then rejects the putative argument he has presented: "This is not an argument against open source, it is an argument against GPL. And if it is a strong argument against GPL, then it is also an argument against the government supporting proprietary projects as well."

We disagree. Software released under a BSD-style license (or into the public domain) can indeed by used readily by others. The owners of proprietary software have profit incentives to make sure that their technology gets used in ways that consumers value. In both of these cases, parties that highly value the results of that R&D can make use of it in their own products. That is not true of software released under the GPL: only other GPL software can make use of it.

The oddity of government sponsorship of R&D for GPL projects can be seen by drawing an analogy with, for example, sponsorship of biotech research. Suppose that the government required that all biotech patents issued under sponsored research be put into a special patent pool. Suppose further that the patents in this pool could be relied on by anyone, for any purpose, with one condition: if products or processes were developed (to any extent whatsoever) under these patents, then all other patents on which these products or processes relied also had to be added to the special patent pool. The obvious outcome of such an arrangement would be that for-profit firms would avoid relying on all patents in the special patent pool—and would donate few, if any, patents to the pool.

VII. Conclusions

We are aware of no general market failure that governments have identified in the provision of commercial software. Yes, software has somewhat unusual characteristics, but so

[238] Lawrence Lessig, *The Future of Ideas*, Random House: New York (2001), pp. 329-330.

do other industries based on intellectual property. And the commercial software industry has grown enormously over the last few decades, providing ever more powerful, easy-to-use software to ever more users. Open-sources software in general has had some successes, but GPL software to date has seen relatively few hits, and those seem mostly imitative.

We are also aware of no compelling evidence that governments have special expertise in analyzing the software industry to effect solutions that will improve the situation. It is perhaps human nature for bureaucrats (and economists) to believe that they can improve upon the operation of markets with strange characteristics. In general, however, we believe that humility is in order. The last 20 years have shown that governments have no particular skill in choosing industries to support as part of "industrial policy" initiatives. We see no reason to believe that governments would be any better at designing new and improved software industries.

It is difficult to know what to make of statements like the following: "Likewise with the government's choice of operating systems. What reason does the government have for supporting closed code, when open code is as powerful and the externalities from using open code would benefit other users?"[239] Whether "open code" in any given situation is actually "as powerful" as "closed code" is an every day business judgment that should be made by businesses, governments, and private users; it does not strike us as a policy issue that should be decided by bureaucrats or legislators, or even by lawyers and economists.

Moreover, we are highly skeptical of the "externalities" claim. To the extent that these "externalities" arise from use (as in the network effects discussed above), we see no reason to believe that they are more important for "open code" rather than "closed code." As discussed above, a switch to increase the "externalities" that benefit other users of open source necessarily decreases the "externalities" of the remaining users of proprietary software—and the net effect is likely to be a reduction in the total "externality" benefits of software.

To the extent that the externalities arise from R&D efforts, then purchase preferences seem to be the wrong tool. Direct support of software R&D (preferably not under the GPL) would be the appropriate policy if such externalities are considered important. Purchase

preferences for open source seem particularly ill-suited for encouraging the development of open source software. Given that the competitive price of open-source software is, in effect, zero, we know of no empirical (or theoretical) evidence that government use of "open code" operating systems will increase basic software R&D.

(...continued)

[239] Lawrence Lessig, *The Future of Ideas*, Random House: New York (2001), p. 247.