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

*David S. Evans**

*Bernard J. Reddy***

Cite as: David S. Evans and Bernard J. Reddy, *Government Preferences for Promoting Open-Source Software: A Solution in Search of a Problem*, 9 MICH. TELECOMM. TECH. L. REV. 313 (2003), *available at* http://www.mttlr.org/volnine/evans.pdf

† © 2002 by David S. Evans and Bernard J. Reddy. All rights reserved.

* 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.

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). This article examines the economic basis for these kinds of government interventions in the market. It first provides some background on the software industry. The article discusses the industrial organization and performance of the proprietary software business and describes how the open-source movement produces and distributes software. It then surveys current government proposals and initiatives to support open-source software and examines whether there is a significant market failure that would justify such intervention in the software industry. The article concludes that the software industry has performed remarkably well over the past 20 years in the absence of government intervention. There is no evidence of any significant market failures in the provision of commercial software and no evidence that the establishment of policy preferences in favor of open-source software on the part of governments would increase consumer welfare.

PART 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 government subsidies of research and development (R&D) for open-source software, standardization on using open-source software, and procurement preferences for 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 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]

1. We use the term "open-source" to refer to software that is made readily available in the form of source code. *See infra* Part II.

2. European Parliament resolution on the existence of a global system for the interception of private and commercial communications (ECHELON interception system) (2001/2098(INI)) (Sept. 5, 2001) [hereinafter European Parliament resolution], *at* http://www.europarl.eu.int/meetdocs/committees/itre/20020325/449496EN.pdf (last visited May 17, 2003).

3. Deutscher Bundestag 14 Wahlperiode [German Bundestag 14th Election Period] (Jun. 20, 2001), http://dip.bundestag.de/btd/14/063/1406374.pdf (last visited May 17, 2003). This legislation differs from the Bundestag's decision in March 2002 to use open source programs such as Linux for some of its own IT needs. See Part VI below.

4. Law on the Establishment of the Agency for Information Technology and Communication in the Administration, Law No. 2001-737 of Aug. 22, 2001, J.O., Aug. 23, 2001, p. 13509 n.194, *available at* http://www.legifrance.gouv.fr/citoyen/jorf_nor.ow?numjo=PRMX0105055D (last visited May 17, 2003).

5. *See*, *e.g.*, Thomas Sterling, *Beowulf Linux Clusters*, *at* http://beowulf.gsfc.nasa.gov/tron1.html (last visited May 17, 2003).

6. *See*, *e.g.*, Press Release, Free Software Foundation, Richard Stallman Inaugurates Free Software Foundation-India, First Affiliate in Asia of the Free Software Foundation (Jul. 20, 2001), *at* http://www.gnu.org/press/2001-07-20-FSF-India.html; Press Release, Free Software Foundation, Richard M. Stallman Addresses Brazilian Congress on Free Software and the Ethics of Copyright and Patents (Mar. 20, 2001) [hereinafter Stallman Addresses Brazilian Congress], *at* http://www.gnu.org/press/2001-03-20-Brazil.txt; Cara Garretson, *Open Source Subject to Fiery Debate*, INFOWORLD, Apr. 12, 2002, http://www.infoworld.com/articles/hn/xml/02/04/12/020412hnopensource.xml.

7. LAWRENCE LESSIG, THE FUTURE OF IDEAS 247 (2001). *See also* Eben Moglen, *Anarchism Triumphant: Free Software and the Death of Copyright*, 4 FIRST MONDAY 8, Aug. 1999,

This article examines the economic basis for these kinds of government interventions in the market. Over 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 among policymakers, 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 recessions. Against this economic backdrop, it is a bit surprising to see so many countries entertaining policies to promote a particular, namely open-source, method for producing and distributing technology.[8] This is surprising because governments have recently been making fewer efforts to endorse particular methods. Although economists often have a myriad of 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,

at http://firstmonday.dk/issues/issue4_8/moglen/index.html; Shawn W. Potter, *Opening Up to Open Source*, 6 RICH. J.L. & TECH. 24 (Spring 2000), *at* http://www.law.richmond.edu/jolt/v6i5/article3.html; 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, *at* http://www.ccic.gov/pubs/pitac/pres-oss-11sep00.pdf (last visited May 17, 2003).

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, Mar. 26, 2002, *available at* http://www.guardian.co.uk/business/story/0,3604,674067,00.html; *Energy Market Trends in The Netherlands 2000*, http://www.ecn.nl/library/reports/2000/p00003.html (last visited May 17, 2003); *Industry suffers as European governments target pharmaceutical spending*, *at* http://www.inpharm.com/External/InpH/1,,1-0-0-0-inp_intelligence_art-0-5523,00.html (Oct. 1, 2001); Ludmila Maksimova, *Russian Government Takes Measures to Regulate Pharmaceutical Market*, *at* http://www.bisnis.doc.gov/bisnis/isa/9902phar.htm (Feb. 1999).

9. *See*, *e.g.*, JOSEPH E. STIGLITZ, PRINCIPLES OF MICROECONOMICS 506 (2d ed. 1997).

which 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).[11] 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.[12] 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.

Part 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. Part 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. Part 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. Part 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. Part VI surveys government programs and proposals to promote open-source software and then

10. *See infra* notes 207 and 208.
11. *See infra* notes 207 and 208.
12. It does not deal with some of the ideological arguments for supporting open source, such as Free Software Foundation's belief that proprietary software owners' negative attitudes about voluntary cooperation "pollute[s] our society's civic spirit." Richard Stallman, *Why Software Should Not Have Owners, at* http://www.gnu.org/philosophy/why-free.html (last updated Feb. 8, 2003).

analyzes these programs and proposals using the market-failure framework discussed above. Part VII presents brief conclusions.

PART 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 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 audio coding patents owned by Fraunhofer IIS-A.[13]

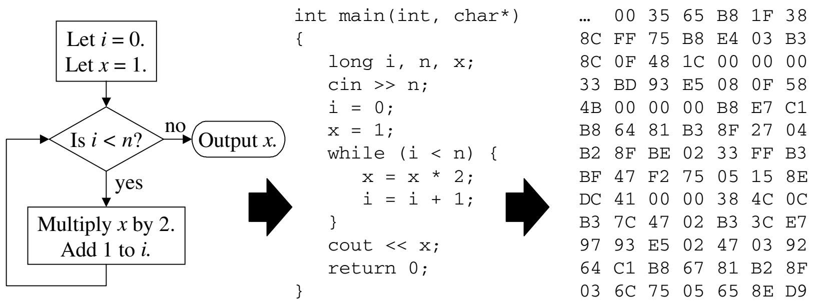
Most programs are written in one of several "high-level" languages. Popular high-level languages include C, C++, Java, Visual Basic, and Pascal. These languages often have commands that look like a written language (usually English) and have meanings that are consistent with written language. For example, "If" and "While" are common commands in many languages. The commands available in high-level languages provide a shorthand for more detailed instructions provided to computers, thus enabling the programmer to avoid many repetitive tasks. 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.[14]

13. *See Mp3licensing.com—About Us—Fraunhofer Gesellschaft*, *at* http://www. mp3licensing. com/about/fhg.html (last visited May 17, 2003).

14. Some languages have interpreters rather than compilers, but the difference is irrelevant for this article.

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 compiled binary code.[15]

FIGURE 1
SIMPLE PROGRAM: FROM DESIGN, TO SOURCE CODE, TO BINARY CODE



```
int main(int, char*)
{
   long i, n, x;
   cin >> n;
   i = 0;
   x = 1;
   while (i < n) {
      x = x * 2;
      i = i + 1;
   }
   cout << x;
   return 0;
}
```

```
…  00 35 65 B8 1F 38
8C FF 75 B8 E4 03 B3
8C 0F 48 1C 00 00 00
33 BD 93 E5 08 0F 58
4B 00 00 00 B8 E7 C1
B8 64 81 B3 8F 27 04
B2 8F BE 02 33 FF B3
BF 47 F2 75 05 15 8E
DC 41 00 00 38 4C 0C
B3 7C 47 02 B3 3C E7
97 93 E5 02 47 03 92
64 C1 B8 67 81 B2 8F
03 6C 75 05 65 8E D9
```

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 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.[16]

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

16. The source would be difficult, but not impossible to determine. 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 Windows applications can be run on Unix-like

Software creators can use various mechanisms to prevent others from using their software and any intellectual property embodied in that software. Patents can protect algorithms and other creative aspects of software design. Copyrights on software code (binary and source) could 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, the most common forms of protection are trade secrets (through distributing the code only in binary form) and copyright.[17]

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 ("proprietary").[18] The second is between distributing software at no direct charge ("unpaid")[19] versus direct charge ("paid") software distribution.[20]

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.[21] Before the

operating systems without having to be rewritten. *See WINE Development HQ—About*, *at* http://www.winehq.com/about.shtml (last visited May 17, 2003).

17. Parties who hold copyrights on open-source software can bring legal action against infringers to enforce the terms of their licenses, as is also true for proprietary software. The Free Software Foundation reports that it has settled many such cases without going to trial. *See* Eben Moglen, *Enforcing the GNU GPL*, *at* http://www.gnu.org/philosophy/enforcing-gpl.html (Sept. 10, 2001). MySQL AB brought claims against NuSphere related to violations of its license terms. MySQL, *FAQ on MySQL v. NuSphere Dispute*, *at* http://www.mysql.com/news/article-75.html (Jul. 13, 2001). These claims were later settled. MySQL Press Release, MySQL AB and Nusphere Corporation Announce Settlement (Nov. 7, 2002), *at* http://www.mysql.com/press/release_2002_14.html.

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

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

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

21. This is somewhat analogous to what many academics do with their writings. Academics try to widely distribute their writings, often incurring production and distribution costs, to gain 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,

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 the binary code without enforcing limitations on further distribution and use. This type of software is usually called "freeware."[22]

The spread of personal computers in homes and offices created a mass market for software. Not surprisingly, many commercial companies emerged to write software for profit. These companies initially protected their investments by enforcing copyrights on 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.[23] The number of software patents awarded annually to U.S. inventors has increased from 829 in 1986 to 7,398 in 2000.[24]

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 $1,880 or $3,135, depending on the platform, and is available for many different kinds of computing platforms.[25] Intuit's QuickBooks accounting package costs between $199 and $500 depending on the edition.[26] Collections of

scientific writings that contain valuable intellectual property are not distributed widely without proper intellectual property protection.

22. The term "shareware" is typically applied to software distributed freely in binary form. However, 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.

23. "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 1986." SAMUEL KORTUM & JOSH LERNER, *Stronger Protection or Technological Revolution: What Is Behind the Recent Surge in Patenting?,* 48 CARNEGIE-ROCHESTER CONFERENCE SERIES ON PUBLIC POLICY 247, 296 (Bennett T. McCallum & Charles I. Plosser eds., Jun. 1998).

24. Number of patents awarded to U.S. inventors with International Patent Classification assignment to subclass G06F ("Electric Digital Data Processing"). Delphion Research, *at* http://www.delphion.com/ (last visited May 13, 2003).

25. *Mathematica: The Way the World Calculates*, *at* http://store.wolfram.com/view/app/mathematica/ (last visited May 17, 2003).

26. *Quickbooks Basic 2003 Overview, at* http://www.quickbooks.com/products/basic/pricing.html (last visited May 17, 2003); *Quickbooks Premier 2003 Overview, at http://quickbooks.intuit.com/qbcom/jhtml/skins/prod_ovw.jhtml?ssaPath=qb_2003_win_premier_1user&productGroup=premier&priorityCode=0273400000* (last visited May 17, 2003).

"utility" programs for Windows (such as Norton Utilities) typically are priced in the range of \$40–\$70.[27] Final Fantasy X, a popular game for Sony's PlayStation 2 console released in 2001, sells for around \$40 a copy;[28] newer video games, such as Splinter Cell, sell for around \$50 a copy.[29]

"Open-source" software is distributed under very different terms than is typical proprietary software. First, the source code (although protected by copyright, as with proprietary software) is made publicly available. Second, the software is distributed under a license that enables people to use the source code only if they comply with certain conditions. The BSD license,[30] perhaps the oldest open-source license, has been modified, but typically allows the free use of the source code so long as the original copyright is acknowledged. 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;[31] the latest version of the Macintosh operating system is also based in part on a BSD version of Unix.[32] Importantly, people can charge for the modified code and need not give away their modifications.[33]

Although many widely used open-source software products have been distributed under a BSD-style license, the General Public License

27. For example, Symantec charges \$50 for the standard version of its antivirus program and \$70 for SystemWorks, a utility suite. Norton Antivirus 2003, *at* http://www.symantecstore.com/dr/sat/ec_MAIN.Entry17c?CID=48782&SID=27674&SP=10007&PN=5&PID=367322&DSP=&CUR=840&PGRP=0&CACHE_ID=48782 (last visited May 17, 2003); Norton SystemWorks 2003, *at* http://www.symantecstore.com/dr/sat/ec_MAIN.Entry17c?CID=48782&SID=27674&SP=10007&PN=5&PID=426992&DSP=&CUR=840&PGRP=0&CACHE_ID=48782 (last visited May 17, 2003).

28. EBGames.com, Final Fantasy X, http://www.ebgames.com/ebx/categories/products/product.asp?pf_id=182336 (last visited May 17, 2003).

29. EBGames.com, Tom Clancy's Splinter Cell, http://www.ebgames.com/ebx/categories/products/product.asp?pf_id=232539 (last visited May 17, 2003).

30. This license was created by the University of California at Berkeley primarily to distribute Unix-related software. PETER H. SALUS, A QUARTER CENTURY OF UNIX 142–143 (1994).

31. *Id.* at 216.

32. *UNIX Based: The Open Desktop, An Open Source Core Lets You Check Out What's Under the Hood*, *at* http://www.apple.com/macosx/technologies/darwin.html (last visited May 17, 2003).

33. *E.g.*, Sendmail, Inc. sells a modified version of its open-source "Sendmail" e-mail server software. Sendmail, *Company Overview*, *at* http://store.sendmail.com/cgi-bin/smistore/company.jsp?BV_UseBVCookie=Yes&filepath=overview/index.shtml&heading=Company%20Overview (last visited May 22, 2003).

(GPL)[34] has become the dominant form of licensing for open-source software.[35] Many of the key persons and institutions spearheading the open-source movement believe that this is the license that will best achieve their goals.[36] If a program is distributed under the GPL, the source code for the program is made available (as is also true with other open-source licenses). 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.[37] That precludes a firm from building a proprietary product based on software licensed under the GPL since 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 relying on software licensed under the GPL are "infected" by the GPL. Linux is perhaps the most popular software released under the GPL.[38]

The remainder of this article primarily considers two major types of software at the center of public-policy debates: proprietary software and open-source software distributed under the GPL.[39] The public policy

34. The GPL is legal language written by the Free Software Foundation and is freely available at http://www.fsf.org for use by others.

35. Hard information on the importance of various licenses is unavailable. As discussed below, SourceForge, a major hosting site for open-source projects, 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 where license information is available, roughly 70 percent rely on the GPL, at least in part. *See* http://sourceforge.net (last visited May 6, 2003).

36. *See infra* Part IV.A.1.

37. Any recipient of a copy of modified code distributed under the GPL can redistribute the code. Since it is impossible to restrict the number of copies that this recipient and subsequent recipients make, there is no effective restriction on supply. The competitive price of software distributed under the GPL, therefore, tends to zero.

38. The key persons and institutions favoring the GPL also favor the term "GNU/Linux" over simply Linux to emphasize that the operating system commonly called "Linux" relies on much more than just the Linux "kernel." Richard Stallman, *What's in a name?*, *at* http://www.fsf.org/gnu/why-gnu-linux.html (last visited May 18, 2003). In keeping with common usage, however, we use the term "Linux" for both the operating system and the kernel.

39. Another type is software that is developed by people or businesses for their own use. Most large companies, and many smaller ones, have written software applications that accomplish particular functions (e.g., accounting). This software is proprietary, but the company typically chooses not to license it to others. Firm-specific software would typically provide little or no benefit to other firms (except perhaps to competitors as a form of market intelligence).

debate centers around them for several reasons: first, most of today's widely used software is proprietary; second, the GPL currently appears to be the most popular license for open-source software; and given the distribution restrictions of the GPL, proprietary software and GPL software (unlike BSD software) face a potentially uneasy coexistence.

PART 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 its changes over time.

Worldwide commercial software[40] is a $171 billion business based on revenue.[41] Revenues from U.S.-based commercial software companies amounted to over $90 billion in 2000.[42] That makes the U.S. commercial software industry larger than the motion picture industry ($74 billion in 2000) and around a fifth of the size of the auto industry ($408.6 billion in 2000).[43] Generally, software is a relatively inexpensive but critical input into the production of computing services. This section describes the structure and performance of the commercial software sector.[44]

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

40. In this section, "commercial software" 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.

41. Richard V. Heinman et al., *IDC Report #25569, Worldwide Software Market Forecast Summary, 2001–2005*, Table 1 (Sept. 2001).

42. *Id.* at Table 4.

43. Sherlene K. S. Lum & Brian C. Moyer, *Gross Domestic Product by Industry for 1998–2000*, Survey of Current Business 30, Table 8, *available at* http://www.bea.doc.gov/bea/ARTICLES/2001/11november/1101gdpxind.pdf (last visited May 18, 2003).

44. For more information regarding the software industry, see Kenneth G. Elzinga & David E. Mills, *PC Software*, 44 ANTITRUST BULL. 739 (1999).

percent.[45] 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.[46] IDC, the leading vendor of data on the software industry, commented in its 2001 software market forecast that, "it is likely that there are more than 10,000 companies competing in the packaged software market . . ."[47] In the United States, the Census Bureau identified almost 9,000 software firms with over 300,000 employees in 1999.[48]

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	$34,519	$34,028	NA	$8,195
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 United States 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,* 6-7, 18 (Apr. 1989); *IDC Report #9358, 1994 Worldwide Software Review and Forecast*, Table 2, pp. 12-31 (Nov. 1994); Richard V. Heinman, Dennis Byron, R. Paul Mason & Melita Marks, *IDC Report #25569, Worldwide Software Market Forecast Summary, 2001 - 2005*, Tables 1, 16, 17, 18 (Sept. 2001); *Bureau of Labor Statistics Data: Consumer Price Index—All Urban Consumers*, http://data.bls.gov/servlet/SurveyOutputServlet (last visited Apr. 18, 2003).

45. Paul Mason et al., *IDC Report #8324, 1993 Worldwide Software Review and Forecast*, Table 2 (Dec. 1993); Heinman, *supra* note 41, at Tables 1, 16.

46. U.S. Census Bureau, *Number of Firms, Number of Establishments, Employment, and Annual Payroll by Employment Size of the Enterprise for the United States, All Industries—1992* [hereinafter U.S. Census Bureau—1992], http://www.census.gov/csd/susb/usalli92.xls (last visited May 18, 2003); U.S. Census Bureau, *Number of Firms, Number of Establishments, Employment, and Annual Payroll by Employment Size of the Enterprise for the United States, All Industries—1999* [hereinafter U.S. Census Bureau—1999], http://www.census.gov/csd/susb/usalli99.xls (last visited May 18, 2003). Figures based on the Bureau's classification of "Prepackaged Software" (NAICS 5112 for data collected in 1997 and later; SIC 7372, prior to 1997).

47. Heinman, *supra* note 41, at 15.

48. U.S. Census Bureau—1999, *supra* note 46.

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: Richard V. Heinman, Dennis Byron, R. Paul Mason and Melita Marks, *IDC Report #25569, Worldwide Software Market Forecast Summary, 2001–2005*, Tables 1, pp. 1-2, 15 (September 2001).

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.[49] According to the latest Census data, over 47 percent of all manufacturing industries have a four-firm concentration ratio greater than that of the software industry.[50]

49. Heinman, *supra* note 41, at 27–29.

50. U.S. Census Bureau, *Concentration Ratios in Manufacturing: 1997 Economic Census* at pp. 7, 16 (Jun. 2001) [hereinafter Concentration Ratios – 1997], http://www.census.gov/prod/ec97/m31s-cr.pdf (last visited May 18, 2003). Figures based upon value of shipments for industries categorized at the 4-digit NAICS industry level.

A second measure of industry concentration is the Herfindahl-Hirschman Index (HHI), widely used in antitrust analysis.[51] 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).[52] In 2000, the HHI for the software industry was 244,[53] a relatively low HHI when compared with other familiar industries such as automobiles (2,506) or breakfast cereals (2,446).[54]

There is also a great deal of turnover among the leading firms, which indicates that firms generally are not entrenched. Five of the top ten companies in 1990 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 decade.[55] 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.[56]

B. *Production Method*

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

51. 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). *See* http://www.usdoj.gov/atr/public/testimony/hhi.htm.

52. 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*, *Section 1.5 Concentration and Market Shares* (revised Apr. 8, 1997), *available at* http://www.usdoj.gov/atr/public/guidelines/horiz_book/15.html (last visited May 18, 2003).

53. Heinman, *supra* note 41, at Table 2, pp. 27–29 (listing figures based on worldwide packaged software revenue). This report provides firm-level data for the 100 largest software vendors which covers 61 percent of packaged software. Firms ranked 80th to 100th largest in size each hold one percent of the market. For purposes of this calculation, we assume 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).

54. Concentration Ratios—1997, *supra* note 50. 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, the same document shows that 55 percent of industries, calculated at the four-digit level, had a higher HHI than the software industry.

55. Mason, *supra* note 45, at Table 2, pp. 10–16; Heinman, *supra* note 41, at Table 2, pp. 27–29.

56. IMS Health, *M&A Drives Decade of Change*, *at* http://www.ims-global.com/insight/news_story/0104/news_story_010425.htm (Apr. 25, 2001).

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 few dollars.[57] Sales, marketing, and promotional expenses can take different forms depending on the type of software and target customer. 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 PC end users is often priced at less than $100 and might be sold through

57. The fat manuals of the past have largely been replaced by electronic documentation, available either on CD or the Web.

retail stores, over the Web, through computer manufacturers, via direct mail solicitation, and so forth.

Support costs can also widely vary, 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 limited support via phone or email. Developers of mass-market software have incentives to design software that has the 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.[58]

After a product ships, two processes often begin: maintenance work on the just-shipped product to fix bugs or add new features; and designing the next version of the software.[59] Games software might not, strictly speaking, have "new versions," but successful games typically have sequels.[60]

One feature of the software industry emerges clearly from this review: average costs of a 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[61] will sometimes be support, sometimes materials and packaging. But for typical PC software, these costs are low compared with the usual development costs.[62]

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

58. For example, PC manufacturer eMachines charges $20 per out-of-warranty support incident; presumably the cost to eMachines of a support call is in the same ballpark. EMachines, *Customer Support*, *at* http://www.emachines.com/support/tech_support.html (last visited May 13, 2003).

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

60. We previously mentioned the game Final Fantasy X, *supra* Part II, the 9th sequel of the first Final Fantasy game.

61. 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).

62. *See, e.g.*, Elzinga, *supra* note 44, at 756.

marginal cost in order to cover its fixed costs.[63] Most software projects are losers in the marketplace, but the financial bonanza available for a winner gives firms incentives to invest.[64] 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.

Because software production has high first costs and low marginal costs, 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. Such characteristics can lead a market to have only a very small number of active suppliers.

This potential for "natural monopoly" is increased if a software category exhibits "network effects." Network effects can arise on either the demand side or the supply side of the market. On the demand side, if most business users of computers use (for example) the same word processing program, then it is relatively easy to trade files, to transfer knowledge of how to use software, and so forth. On the supply side, the availability of complements can give rise to network effects. The more applications existing 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; these network effects provide substantial benefits to users, enabling a winning program to make more sales. But category leaders can and do get replaced.[65] In a software category with strong network

63. This may not be true for every product, especially if a firm produces complementary products. 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. See AOL Time Warner, 2002 Annual Report 25–27, http://www.aoltimewarner.com/investors/annual_reports/pdf/2002ar.pdf (May 18, 2003).

64. 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 463 (David S. Evans ed., 2002).

65. *See* DAVID S. EVANS, ALBERT NICHOLS, & BERNARD REDDY, *The Rise and Fall of Leaders in Personal Computer Software*, *in* MICROSOFT, ANTITRUST AND THE NEW ECONOMY: SELECTED ESSAYS 265, 267 (David S. Evans ed., 2002); STAN J. LIEBOWITZ & STEPHEN E.

effects, competition is typically dynamic competition *for* the market, rather than static price competition *within* the market.[66] Firms compete by coming out with the best, most attractive new products, thereby attracting the bulk of the customers, not by dropping prices for existing products. As a result, the existence of a "dominant" firm in a software category does not imply that some type of "market failure" exists that government intervention can (and should) try to fix. But note that not all software categories exhibit network effects. The existence of heterogeneous groups of customers (i.e., customers with different preferences) can enable multiple software firms to coexist in the same category, with different firms targeting the preferences of the different groups of customers.

Commercial firms typically protect their intellectual property using copyrights, patents, and trade secrets, as discussed above. For instance, they license only binary code to make reverse engineering more difficult for competitors, thus making it harder to copy key program features. They take these steps to protect their opportunity to recover their fixed investment cost or to be rewarded for the risks borne in financing software creation and development. 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. 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

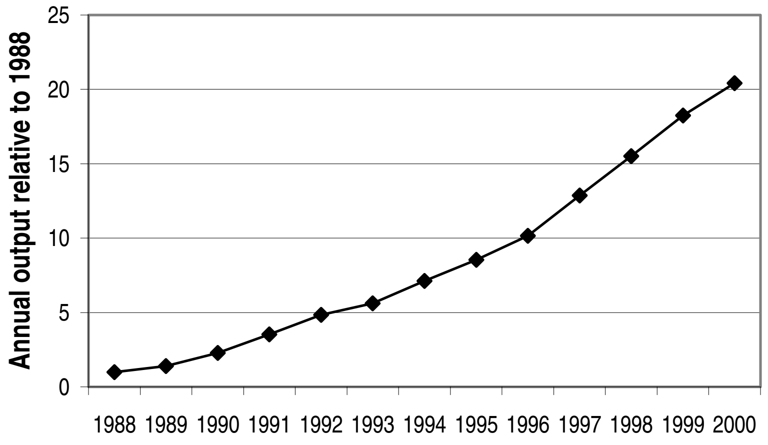
The commercial software industry has exploded since its early years. Quality-adjusted prices have fallen markedly in nominal terms, even before adjusting for inflation. Quality-adjusted output has soared. By any measure, the commercial software industry is succeeding in providing ever-more powerful products 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

MARGOLIS, WINNERS, LOSERS & MICROSOFT: COMPETITION AND ANTITRUST IN HIGH TECHNOLOGY (rev. ed. 2001).

66. *See* David S. Evans & Richard Schmalensee, *Some Economic Aspects of Antitrust Analysis in Dynamically Competitive Industries*, *in* INNOVATION POLICY AND THE ECONOMY 2 (Adam B. Jaffe et al. eds., 2002).

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

FIGURE 2
WORLDWIDE QUALITY-ADJUSTED PACKAGED
SOFTWARE SALES, 1988–2000



Sources: Paul Mason et al., IDC Report #8324, *1993 Worldwide Software Review and Forecast*, Table 2, pp. 10–16 (Dec. 1993); Evan Quinn et al., *IDC Report #10460, 1995 Worldwide Software Review and Forecast*, Table 2, pp. 15–30 (Nov. 1995); Steve Garone et al., *IDC Report #12408, 1996 Worldwide Software Review and Forecast*, Table 2, pp. 17–33 (Nov. 1996); Jacqueline Sweeney et al., *IDC Report #14327, 1997 Worldwide Software Review and Forecast*, Table 2, pp. 30–47 (Oct. 1997); Steve McClure et al., *IDC Report #20161, 1999 Worldwide Software Review and Forecast*, Table 3, pp. 36-38 (Oct. 1999); R. Paul Mason et al., *IDC Report #22766, Worldwide Software Market Forecast Summary, 2000–2004*, Table 3, pp. 43-45 (Aug. 2000); Richard V. Heinman et al., *IDC Report #25569, Worldwide Software Market Forecast Summary, 2001 - 2005*, Table 2, pp. 27–29 (Sept. 2001); Bureau of Economic Analysis, and the U.S. Department of Commerce, *2001 Annual NIPA Revision*, Tables 1, 11 (Aug. 2001), *available at* http://www.bea.doc.gov/bea/papers/tables.pdf (last visited Feb. 7, 2003); Robert Parker & Bruce Grimm, *Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts, 1959–98*, Bureau of Economic Analysis and the U.S. Department of Commerce, *available at* http://www.bea.doc.gov/bea/papers/software.pdf (last visited Jan. 31, 2003).

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.[67] 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 50-fold from 1983 to 2000 (data not shown).[68]

b. Price and Performance Have Improved Markedly

The Bureau of Labor Statistics publishes a Consumer Price Index (CPI) for "computer software and accessories" (available from December 1997 onward).[69] 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

67. Robert Parker & Bruce Grimm, *Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts, 1959–98*, Bureau of Economic Analysis and the U.S. Department of Commerce, http://www.bea.doc.gov/bea/papers/software.pdf (last visited May 18, 2003). The BEA constructed this price index for "prepackaged" software by splicing together 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. *Id.* at 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. *Id.* at 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, with changes in the own-account software price index receiving a 75 percent weight. *Id.* at 17.

68. *IDC Report #4046*, *1989 Software Review and Forecast,* 6–7, 18 (Apr. 1989); Heinman, *supra* note 41, at Table 1, pp. 1–2.

69. Bureau of Labor Statistics Division, *Consumer Price Index—All Urban Consumers: Computer Software and Accessories*, http://data.bls.gov/labjava/outside.jsp?survey=cu (last visited May 6, 2003). To obtain CPI data, select the appropriate options—in this case, "U.S. City Average" and "Computer software and accessories."

9.5 percent.[70] This means that the real price of software fell by approximately 27.4 percent over this period.[71]

Computer hardware and software both have 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.[72] Based on system price relative to performance over that same time period, the most efficient system available improved by a factor of 12.[73] 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.[74] By 2000, that number increased to 10 percent.[75] As mentioned above, patenting of software has

70. Bureau of Labor Statistics Division, *Consumer Price Index—All Urban Consumers: All Items* (1997–2001), http://data.bls.gov/servlet/SurveyOutputServlet?jrunsessionid=1043186181547150272 (last visited Jan. 21, 2003). In this case, select "All items" instead of "Computer software and accessories."

71. To buy the equivalent of one dollar's worth of goods in December 1997 ("all items") in December 2001 would cost $1 + (9.5% × $1) = $1.095. However, to buy one December 1997 dollar's worth of software in 2001 would cost $1 + (-20.5% × $1) = $0.795. $0.795 / $1.095 = $0.726, so the real price of software—*i.e.*, the price of software relative to "all items"—fell by approximately ($1–$0.726) / $1 = 27.4 percent.

72. 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, whose results are publicly available. The benchmark measures both performance and price/performance. *At* Transaction Processing Performance Council, *Complete TPC-C Results List – Sorted by HardwareVendor, Version 3 Results,* http://www.tpc.org/tpcc/results/tpcc_results.asp?print=true& OrderBy=&version=3 (last visited May 13, 2003).

73. *See id*.

74. Software firms are those that report their primary SIC code to be 7372 (software publishers). Standard & Poor's Compustat®, *available at* http://www.compustat.com/www (last visited May 13, 2003).

75. Software firms are those that report their primary SIC code to be 7372 (software publishers). *Id.*

also increased substantially.[76] In 1986, 829 patents were granted for software with at least one U.S. inventor. In 2000, 7,398 patents were granted.[77] 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.

a. Software Production in the Mainframe Era

The 1960s and 1970s were the era of expensive mainframe and minicomputers.[78] 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.[79] These products were typically expensive, although much

76. Some commentators have argued that it should not be possible to obtain patents on software. LESSIG, *supra* note 7, at 205–17. 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." *Id.* at 206.

77. *See* Delphion Research, *supra* note 24.

78. *See* PAUL FREIBERGER & MICHAEL SWAINE, FIRE IN THE VALLEY (2d ed. 2000); STEVEN LEVY, HACKERS (1984); STEPHEN MANES & PAUL ANDREWS, GATES (1993) (discussing the state of computer hardware and software in the years just before the emergence of the personal computer). *See also* The Software History Center, *at* http://www.softwarehistory.org (last visited May 18, 2003).

79. In the PC era, MultiMate had a similar origin; its original developer first wrote a look-alike of the Wang word processor for an insurance company and then marketed the software generally. *See* Ken Polsson, *Chronology of Events in the History of Microcomputers*, *at* http://web.archive.org/web/19990427035916/http://www.islandnet.com/~kpolsson/comphist/comp1981.htm (last updated Feb. 3, 1999).

less expensive (but possibly less versatile) than custom software.[80] 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, such as the authors, since the mid-1970s.[81]

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.[82]

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.[83] 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.[84] Bill Gates and Paul Allen wrote their first version of MS-BASIC and formed Microsoft in 1975.[85] Word processing programs (such as Electric Pencil and later WordStar)[86] 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.[87] Database programs such as dBase II came along a few years

80. One veteran of that era has described starting a firm in 1971 to sell a payroll system and making 20–30 sales per year. Luanne Johnson, *From Not-Invented-Here to Off-The-Shelf* (1997), http://www.softwarehistory.org/history/Johnson2.html (last visited May 13, 2003).

81. *See* SAS corporate web site, *Company/*, *at* http://sas.com/corporate/index.html (last visited May 13, 2003); SPSS corporate web site, *SPSS Inc. Corporate History*, *at* http://spss.com/corpinfo/history.htm (last visited May 13, 2003).

82. Richard Stallman, Address at New York University, Free Software: Freedom and Cooperation (May 29, 2001), *available at* http://www.fsf.org/events/rms-nyu-2001-transcript.txt.

83. As a legal matter, software is generally "licensed" for use rather than "sold," with the terms of the license determining how the buyer can use the software. Much the same is true of, for example, music CDs; someone who purchases a music CD generally does not obtain rights to copy the CD for sale, broadcast the music, and so forth.

84. Polsson, *supra* note 79.

85. FREIBERGER, *supra* note 78, at 53–54 (explaining that the first version of MS-BASIC was for the MITS Altair). *See also* Microsoft Museum, *Microsoft Timeline*, *at* http://www.microsoft.com/museum/musTimeline.mspx (last visited May 13, 2003).

86. FREIBERGER, *supra* note 78, at 187, 194.

87. *Id.* at 289–291.

later.[88] While early PC users did swap software—Bill Gates raised the ire of hackers by complaining about their pirating of MS-BASIC[89]—there was no organized open-source movement, and the open-source approach played no visible role in the early history of PC software.

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. Operating systems and applications software increasingly came from firms independent of the computer vendors. 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 (first CP/M, then MS-DOS, and later Windows), however, were licensed by software firms to 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. This shift away from a vertically integrated approach was a marked change from the prior mainframe/minicomputer world.

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.

PART 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

88. Polsson, *supra* note 79.
89. FREIBERGER, *supra* note 78, at 195.

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 does share 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[90]—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."[91] 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."[92]

Stallman and the FSF began work on the building blocks for an operating system in the 1980s. This effort started with development tools such as editors (like Emacs)[93] and compilers (like the GCC).[94] 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[95] that have no parallel available to proprietary software, they will provide a range of useful modules to serve as building blocks in

90. GNU stands for "GNU's Not Unix."

91. Free Software Foundation, *The Free Software Definition*, *at* http://www.fsf.org/philosophy/free-sw.html (last visited May 18, 2003).

92. Richard Stallman, *The GNU Project*, *at* http://www.fsf.org/gnu/thegnuproject.html (last visited May 13, 2003).

93. *Id.*

94. *Id.*

95. In this context, a "library" is a piece of software that is intended to provide services to some other software; it can be considered a building block for a program.

> 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.[96]

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:

> 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.[97]

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":[98]

> 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.[99]

96. Richard Stallman, *Why You Shouldn't Use the Library GPL for Your Next Library*, *at* http://www.fsf.org/philosophy/why-not-lgpl.html (last visited May 13, 2003).

97. Richard Stallman, *The GNU GPL and the American Way*, *at* http://www.gnu.org/philosophy/gpl-american-way.html (last visited May 13, 2003).

98. *See* Free Software Foundation, *What Is Copyleft?, at* http://www.fsf.org/copyleft/copyleft.html (last visited May 13, 2003).

99. The GPL can affect patent and copyright rights. If a firm (or programmer) has patent rights to 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,

The copyleft provision means that if people choose to distribute software that is based 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 makes it essentially impossible for anyone to develop proprietary commercial software using code subject to the GPL.[100]

Importantly, there is some ambiguity over when "contact" with a program licensed under the GPL infects other work. That is because the phrases "work . . . derived from the Program"[101] and "based on the Program" are not clear. The license explicitly states 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.[102] 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.[103]

GNU General Public License, *at* http://www.fsf.org/copyleft/gpl.html (last visited May 13, 2003).

100. Explicit permission to develop, however, may be obtained from the copyright holder. The program's copyright holder 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. Dual-licensing does occur, but the FSF warns the public that it rarely permits dual licensing of software whose copyright it owns. *See* Free Software Foundation, *Frequently Asked Questions about the GNU GPL*, *at* http://www.fsf.org/copyleft/gpl-faq.html (last visited May 13, 2003). *See also id.* at *Using a certain GNU program under the GPL does not fit our project to make proprietary software. Will you make an exception for us? It would mean more users of that program.* Dual licensing becomes difficult, if not impossible, to arrange if a GPL program has gone through many generations and has multiple copyright owners.

101. "Program" refers to the program being licensed under the GPL. *See GNU General Public License*, *supra* note 99.

102. 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.

103. Free Software Foundation, Frequently Asked Questions, *supra* note 100. *See also id.* at *What is the difference between 'mere aggregation' and 'combining two modules into one*

The terms of the GPL apply only to the distribution of software licensed under the GPL, although the definition of "distribution" in this context is unclear. An enterprise may be able to modify a GPL program and use it internally without being legally bound to make the modified source code 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.[104]

B. *Production of Open-Source Software*

Open-source software has primarily been developed by individuals who donate their time to work on projects that interest them. The original developers typically begin work on idea that they find interesting, useful, or both. They eventually solicit support from other interested programmers, often communicating over the Internet. Programmers, including the original developers, may come and go during the course of the project as they complete work and as their interests wax or wane. A core group, often consisting of one or more of the original developers, has the responsibility of suggesting and incorporating changes. Modified versions of the source code are posted on the Internet and available for free to anyone who wants to use or modify it further. Over time, users may end up running the software with different 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. Programmers may ask themselves "what would I like my software to do?" which may then be augmented by self-selected user feedback. Second, there is little extensive, formal testing of the type that commercial firms often must engage in.[105] Testing is instead performed in uncontrolled environments,

program'? and *If a program released under the GPL uses plug-ins, what are the requirements for the licenses of a plug-in?*.

104. Ira V. Heffan, *Copyleft: Licensing Collaborative Works in the Digital Age*, 49 STAN. L. REV. 1487 (1997); David McGowan, *Legal Implications of Open-Source Software*, *at* http://papers.ssrn.com/paper.taf?abstract_id=243237 (last visited May 13, 2003).

105. Formal commercial internal testing might include using hundreds, perhaps thousands of hardware/software configurations in a controlled manner.

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. Some might consider the lack of structure a benefit, since innovations can come from anywhere, while others might consider it a potential hindrance, since 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 differ from those for proprietary software. Understanding these incentives is important in order to make 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.[106] 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.

106. *See*, *e.g.*, Josh Lerner & Jean Tirole, *Some Simple Economics of Open Source Software*, 50 J. INDUS. ECON. 197, 197–234 (2000); Justin Pappas Johnson, Some Economics of Open Source Software (Dec. 11, 2000) (unpublished paper presented at 2001 IDEA Toulouse conference), *available at* http://www.idei.asso.fr/Commun/Conferences/Internet/Janvier2001/Papiers/Johnson.pdf (last visited May 18, 2003); ERIC RAYMOND, *The Magic Cauldron*, *in* THE CATHEDRAL AND THE BAZAAR (2001), *available at* http://catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/ (last visited May 18, 2003).

- It "scratches an itch."[107] Programmers attack problems that they personally face[108] 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.[109]

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

Open-source advocates like Eric Raymond generally dismiss the possibility that financial rewards are the motivations,[110] although Raymond does acknowledge "it can get you a better job offer, or a consulting contract, or a book deal."[111] This is similar to the signaling incentive that Lerner and Tirole explore.[112] But according to Raymond, this sort of opportunity "is at best rare and marginal for most hackers."[113]

107. "Every good work of software starts by scratching a developer's personal itch." ERIC RAYMOND, *The Cathedral and the Bazaar, in* THE CATHEDRAL AND THE BAZAAR 23 (2001), *available at* http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ (last visited May 18, 2003).

108. 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.

109. 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 "TalkBack" comments on articles about Microsoft posted at http://www.zdnet.com/.

110. Eric Raymond is sometimes considered the resident anthropologist of the open-source movement. *See An Interview with Eric Raymond*, *at* http://opensource.oreilly.com/news/raymond_0101.html (Jan. 24, 2001). 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://catb.org/~esr/writings/(last modified Nov. 23, 2002). For Raymond's commentary on some internal Microsoft documents about open source (the "Halloween Documents"), see http://www.opensource.org/halloween/ (last visited Feb. 5, 2003).

111. ERIC RAYMOND, *Homesteading the Noosphere*, *in* THE CATHEDRAL AND THE BAZAAR 79 (2001), *available at* http://catb.org/~esr/writings/cathedral-bazaar/homesteading/ (last visited May 18, 2003).

112. Lerner, *supra* note 106.

113. RAYMOND, *supra* note 111.

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 are discussed below. 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.

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 develops (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.[114] Any number of firms can have incentives to *use* open-source software. For example, a firm deciding what Web hosting software to use 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.

114. 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, Aug. 24, 2001, *at* http://archive.infoworld.com/articles/op/xml/01/08/27/010827oppetreley.xml.

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.

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,"[115] 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.[116]

We now examine some of the business models that Eric Raymond has identified as possible ways that for-profit firms can try to make money by developing open-source software.[117] For each of these business models, we identify firms that appear to have tried to use them; we also assess the degree to which they have succeeded or failed.

115. Bruce Perens, *The Open Source Definition: Version 1.9*, *at* http://www.opensource.org/docs/definition.html (last visited May 18, 2003).

116. Bruce Perens, *Open Letter to Michael Robertson*, *at* http://www.lwn.net/daily/perens-robertson.php3 (Apr. 13, 2002).

117. ERIC RAYMOND, *The Magic Cauldron: Indirect Sale-Value Models*, *in* THE CATHEDRAL AND THE BAZAAR 136 (2001), *available at* http://www.catb.org/~esr/writings/magic-cauldron/magic-cauldron.html (last visited May 6, 2003).

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).[118] For example, a business could promote open-source client software for desktops or laptops 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 complementary commercial product's profits must be sufficient to compensate for the cost and risk of the investment in the open-source project.

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 support the development of GPL software in conjunction with their own proprietary software. In addition to IBM, discussed below under hardware vendors, the most public of the firms involved in developing GPL software have included Corel, Ximian, theKompany.com, and MySQL.[119]

Corel came out with its own Linux distribution and developed installation and other tools to make it easy to install. It also aided the WINE project[120] while adapting (or "porting") its proprietary 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.[121] As Perens

118. *Id.*

119. Some firms have supported the development of non-GPL open-source software, typically attempting to license an enhanced version of the software under proprietary terms—for example, Sendmail, Inc. *See* Sendmail, *Company Overview*, http://store.sendmail.com/cgi-bin/smistore/company.jsp?BV_UseBVCookie=Yes&filepath=overview/index.shtml&heading=Company%20Overview (last visited May 6, 2003).

120. 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 (the Lesser General Public License), 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*, *at* http://linux.corel.com/support/wine_faq.htm (on file with author); Alexandre Julliard, *Conclusions*, *at* http://www.winehq.com/hypermail/wine-license/2002/03/0029.html (Mar. 4, 2002); Stallman, *supra* note 96.

121. The latest version of its office suite for Linux is WordPerfect Office 2000; the latest version for Windows is WordPerfect Office 2002. Corel Corp., *Products*, *at* http://linux.corel.com/products (on file with author); Corel Corp., *WordPerfect Office 2002 Standard*, *at* http://www3.corel.com/cgi-bin/gx.cgi/AppLogic+FTContentServer?GXHC_GX_

explained, it seems Corel was unable to make money by selling Linux; it did no better in trying to sell WordPerfect for Linux. Thus, this effort must be tagged as a failure.

Ximian has developed a GPL program called Evolution that runs on Linux;[122] it has also been a major contributor to other open source projects like GNOME, a GPL "desktop" user interface for Linux and other Unix-like operating systems. Evolution mimics the operation of Microsoft Outlook, which provides e-mail and other capabilities. Ximian offers a proprietary program that allows Evolution to work with Microsoft's e-mail server, thereby gaining the same types of capabilities on Linux that are available to users of Outlook.[123] 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).[124] 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.[125] The firm also has announced, however, that it will not use the GPL for future products because of the lack of profits.[126] 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.[127] In part, MySQL earns revenues by providing support services to its database users (services that, in

jst=5a964475662d6165&GXHC_gx_session_id_FutureTenseContentServer=355c17803d3e054a&pagename=Corel/Product/Highlight&id=CC1MSNDMYJC&highlight=requirements (on file with author); Press Release, Xandros Announces Strategic Licensing Agreement with Corel Corporation, (Aug. 29, 2001), *available at* http://www.xandros.net/release1.html (last visited May 18, 2003).

122. Ximian, Inc., *Ximian Evolution 1.2*, *at* http://www.ximian.com/products/evolution/ (last visited May 18, 2003).

123. Ximian, Inc., *Ximian Connector for Microsoft Exchange*, *at* http://www.ximian.com/products/connector/faq.html (last visited May 18, 2003).

124. TheKompany.com, *Kivio mp*, *at* http://www.thekompany.com/products/kivio/ (last visited May 18, 2003).

125. *Id.*

126. Shawn Gordon, *Running a Corporation in an Open-Source World*, *at* http://www.linuxandmain.com/essay/sgordon.html (Mar. 18, 2002).

127. Earlier versions of MySQL were released under an even more restrictive license than the GPL. MySQL AB, *MySQL Free Public License Version* 4, Mar. 5, 1995, *at* http://www.mysql.com/support/arrangements/mypl.html (last visited May 13, 2003).

principle, could be provided by any third party who sufficiently studied the product). But MySQL also earns revenues by licensing its flagship product (i.e., exactly the same program that it licenses under the GPL) to firms that want to combine their software with MySQL's in ways that would not be permitted under the GPL. MySQL can follow such a mixed licensing strategy because it owns the copyright on its software. To date, MySQL appears to be successful.[128]

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, but not all, of the software, is under the GPL.[129] 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.[130] IBM's hardware business is unusual because it markets several fundamentally different types of servers with mutually incompatible operating systems.[131] 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 because it

128. Press Release, MySQL AB, Record Sales, New Financing Fuel Growth at MySQL (Apr. 22, 2002), *at* http://www.mysql.com/news/article-96.html. 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*, *at* http://forum.divdata.net/Centera/Documents/Centera/ReiserFS.pdf (last visited May 18, 2003).

129. 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. IBM, *Open AFS Overview*, *at* http://www-124.ibm.com/developerworks/oss/afs/info.html (last visited May 18, 2003); IBM, *Journaled File System Technology for Linux*, *at* http://www-124.ibm.com/developerworks/oss/jfs/index.html (last visited May 18, 2003).

130. James Evans, *IBM to invest almost $1 billion in Linux development*, INFOWORLD, Dec. 12, 2000, *available at* http://archive.infoworld.com/articles/hn/xml/00/12/12/001212hnibmlin.xml.

131. IBM's server types include: mainframes running zOS, midrange servers running OS/400, RISC processor-based Unix servers running AIX, and Intel processor-based servers running Windows. *See* IBM, *IBM E-server*, *at* http://www-132.ibm.com/content/home/store_IBMPublicUSA/en_US/eServer/eServer.html (last visited May 6, 2003).

is also one of the world's largest software vendors.[132] IBM uses Linux (and other open-source software) to help promote sales of both its hardware and 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 libraries, test suites, programming tools, and experimental code.[133] 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.[134] In addition, Hewlett Packard has recently supported the development of open source kernel extensions such as openMosix for the IA64 microprocessor family.[135] And Hewlett Packard also "sponsors" or is a member of a variety of open source organizations.[136] Some other vendors have announced support for open source in more modest ways.[137]

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

132. 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 IBM, *DB2 for Linux*, *at* http://www-3.ibm.com/software/data/db2/linux/ (last visited May 13, 2003); IBM, *Lotus Domino*, *at* http://www.lotus.com/home.nsf/welcome/domino (last visited May 13, 2003).

133. See Intel, *Open Source from Intel*, *at* http://developer.intel.com/software/products/opensource/ (last visited May 18, 2003).

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

135. Moshe Bar, *Porting Linux to the IA64 Platform*, Byte.com, May 13, 2002, *at* http://www.byte.com/documents/s=7182/byt1021067742738/0513_moshe.html (last visited May 7, 2003).

136. HP U.S., *hp and open source*, *at* http://www.opensource.hp.com/ (last visited May 7, 2003).

137. 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. *See* GNOME Foundation Press Releases, The GNOME and Linux Communities and Industry Leaders Join to Create the GNOME Foundation (Aug. 15, 2000), http://www.gnome.org/pr-foundation.html. Sun has continued to work with the GNOME Foundation, and recently contributed to GNOME's Accessibility Framework. *See* GNOME Foundation Press Releases, Making GNOME Accessible—Opening New Doors at the Workplace for Users with Disabilities (Aug. 28, 2001), http://www.gnome.org/pr-accessible.html.

dimension on which they have tried to do that. The "Unix wars" of the 1980s are a good example; competing hardware vendors (IBM, Sun, SGI, Digital, Hewlett-Packard, and others) offered proprietary, and deliberately 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 like Unix. But a standard version of Linux might result in problems for hardware companies. It 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 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 writing open source code, by doing so it may obtain specialized knowledge about open source that could be helpful in providing better services. Or, it might obtain credibility about open source that would improve its ability 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. Cygnus's familiarity with these compilers may have given it a cost advantage over other firms in doing so. Cygnus Solutions was purchased by Red Hat, a Linux distribution company, in 1999.[138]

Red Hat and other Linux distributors arguably sell 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."[139] However, there appear to be few barriers to entry to such a business; with low entry barriers, it is difficult both to make a profit and to fund the development of Linux or other open-source products. This is particularly true when other firms can (and do) copy what (for example) Red Hat

138. *See* Red Hat Network, *Red Hat to Acquire Cygnus and Create Global Open Source Powerhouse* (Nov. 15, 1999), http://www.redhat.com/about/presscenter/cygnus_1999/redhat-cygnus111599.html.

139. RAYMOND, *supra* note 117, at 113, 137.

chooses to distribute,[140] thus severely limiting what Red Hat can charge for its distribution. Several Linux distributions have reported rocky financial results within the last year, and even Red Hat cannot be considered a resounding financial success.[141]

Several firms that support open source 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.[142] Eazel, which developed a file management tool under the GPL, no longer exists.[143] Caldera, a Linux vendor that is not entirely happy with the GPL and sells proprietary software as well as Linux, offers a similar service.[144] Entry into an industry like this does not seem particularly difficult, again making it difficult to both cover costs and fund open-source development.

D. *The Performance of Open Source*

Open-source production methods have created several software products that are widely used 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

140. 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. *See* Daniel Christie, *Linux Mandrake 8.0*, http://www.thedukeofurl.org/reviews/misc/mandrake80/printable.shtml (on file with author).

141. *See, e.g.,* Mandrake Linux, *Mandrake Linux Users Club Answers*, *at* http://www.mandrakelinux.com/en/club/club-answers.php3 (Apr. 4, 2002); Stephen Shankland, *SuSE Linux Gets New CEO, Cuts Staff*, CNET NEWS.COM, Jul. 23, 2001, *at* http://news.com.com/2102-1001-270372.html (last visited May 18, 2003). Red Hat projected that it would lose $140 million over fiscal year 2002 on $79 million in total revenue. *See* Red Hat, Inc. *Consolidated Statements of Operations—FY2002*, http://media.corporate-ir.net/media_files/NSD/RHAT/reports/GAAP_Statement_Ops_FY2002.pdf (Dec. 31, 2001).

142. *See* Red Hat Network, *Overview of Services*, *at* https://rhn.redhat.com/feature_comparison.pxt (last visited Apr. 5, 2001); Ximian, *Ximian Red Carpet: Automated Software Maintenance and Version Management*, *at* http://www.ximian.com/products/ximian_red_carpet/ (last visited May 18, 2003).

143. Peter Galli, *Linux Specialist Eazel Calls It Quits*, *at* http://www.zdnet.com/eweek/stories/general/0,11011,2761184,00.html (May 15, 2001); Linux Today, *Eazel's Demise Is Official*, *at* http://linuxtoday.com/mailprint.php3?action=pv& ltsn=2001-05-16-004-20-NW-GN (May 15, 2001).

144. *See* Mary Jo Foley, *Caldera CEO: MS right to shun open source*, CNETASIA, May 10, 2001, *at* http://asia.cnet.com/newstech/systems/0,39001152,2120279,00.htm (last visited Jan. 20, 2003); SCO, *Caldera Volution Online*, *at* http://www.caldera.com/products/volutiononline/ (last visited May 18, 2003).

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.

a. GPL Successes

The MySQL database and Samba, software that lets servers running Linux and other Unix-like operating systems emulate Windows servers, have been widely praised.[145] Linux, however, is probably the most famous GPL software. Linux is an operating system widely popular on server computers but, at least relatively, much less so on client computers.[146] "Clustering" software for Linux, creating relatively inexpensive "supercomputers," might also be judged a success.[147] 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. Although the interfaces are somewhat like those for Windows and the Macintosh, they are not yet as user-friendly. These desktops are not widely used because of Linux's relative lack of success on client computers, but they may have promise if they and Linux continue to improve for use by non-technical users.

145. *See* MySQL AB Press Release, Leading European VCs Invest in MySQL AB (Nov. 12, 2001), *at* http://www.mysql.com/news/article-84.html; Michael Vizard & Steve Gillmor, *CEO of MySQL explains why open-source may be right approach for many enterprises*, INFOWORLD, Dec. 20, 2001, *at* http://archive.infoworld.com/articles/hn/xml/01/12/20/011220hnmickos.xml; 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, *Samba Survey March '96*, http://www.samba.org/samba/survey/ (May 5, 1997); eWeek, *i3 Awards Finalist*, http://www.eweek.com/print_article/0,3668,a=26153,00.asp (Apr. 29, 2002).

146. 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. Al Gillen, *IDC Report #26952, Worldwide Linux Operating Environments Forecast, 2002–2006: Client Shipments Pick Up the Pace*, Table 4, p. 8 (Feb. 2002).

147. Rick Cook, *Supercomputers on the Cheap*, LINUXWORLD, Apr. 13, 2000, *at* http://www.cnn.com/2000/TECH/computing/04/13/cheap.super.idg/.

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 BSD Unix that are 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.[148] Apple's user interface is not open source, but the BSD Unix that underlies it is open source.[149] Two other key pieces of software emerged from work done at Berkeley roughly two decades ago: BIND and Sendmail.[150] 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).[151] Sendmail was the first modern e-mail server software and is still widely used.[152]

Other widely-used open-source projects also have BSD-style licenses. Three such pieces of software are Apache, Xfree86, and Perl. Apache, described previously, is perhaps the most widely used Web server software.[153] Xfree86 is an implementation of a windowing system for Linux and other Unix-like operating systems for Intel-compatible computers. Xfree86 seems to be included with essentially all Linux distributions and is usable on both clients and servers. Perl is a scripting language widely used on servers and on clients by developers.[154]

148. *See* Mac, *Mac OS X System Architecture*, *at* http://developer.apple.com/macosx/architecture/index.html (last visited May 18, 2003).

149. *Id.*

150. Nominum, *BIND (Berkeley Internet Name Domain)* (2001) (on file with author); *Red Hat Sendmail HOWTO: 1.2 History*, http://www.redhat.com/support/resources/howto/RH-sendmail-HOWTO/x29.html (last visited May 6, 2003).

151. *See* Internet Software Consortium, *ISC BIND*, *at* http://www.isc.org/products/BIND/ (last visited Jan. 20, 2003).

152. Sendmail, *Company Overview*, *at* http://store.sendmail.com/cgi-bin/smistore/company.jsp?BV_UseBVCookie=Yes&filepath=overview/index.shtml&heading=Company%20Overview (last visited May 18, 2003).

153. Apache, *HTTP Server Project*, *at* http://httpd.apache.org/ (last visited May 6, 2003).

154. 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.

c. Some Failures[155]

In general, open source has not been successful in developing user-friendly software aimed at mass-market users. Office suites have generally not succeeded. This may change since new suites are under development in conjunction with both the KDE and GNOME desktops, but neither can yet be considered a success.[156] The best of the existing open-source office suites is probably OpenOffice.[157] OpenOffice is based on the source code for StarOffice, a proprietary product acquired by Sun. StarOffice's use was too tiny for market research firms to measure when Sun acquired its developer.[158] Games are a large software category for PCs and game consoles, but we are unaware of any commercial-quality games developed under the GPL.

2. Open-Source Projects under Development

A large number of open-source projects are currently under development. Most of these projects seem to rely on the GPL rather than licenses with fewer commercialization restrictions. SourceForge is a Web site that provides free hosting services to open-source projects.[159] In early May 2002, 38,610 projects were hosted on SourceForge.[160] Of these,

155. "Failures" is defined as failures for open source as a whole, not individual product failures.

156. 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. *See GNOME Office*, *at* http://www.gnome.org/gnome-office/index.shtml (Jan. 23, 2003).

157. 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*, WASH. POST, May 12, 2002, at H07, *available at* http://www.washingtonpost.com/ac2/wp-dyn/A4246-2002May11?language=printer.

158. 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 (Mar. 19, 2002), *at* http://www.sun.com/smi/Press/sunflash/2002-03/sunflash.20020319.1.html (last visited May 18, 2003).

159. See Sourceforge, *supra* note 35.

160. Around May 6, 2002, researchers under our direction executed a computer program that automatically downloaded 38,610 Web pages from the SourceForge site. Our researchers then automatically extracted from those Web pages a variety of information about the 38,610 projects hosted by SourceForge and stored that information in a database. The information presented in this paragraph and the next was obtained from queries against that database.

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 (the Lesser General Public License), a cousin to the GPL, also favored by the FSF for some purposes.[161] 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.

SourceForge provides other useful information (at least for projects hosted at that site) on the types of open-source projects currently under development and who the intended audiences are for these projects. Of the 25,852 projects whose development status is known, only 4,712 (18 percent) had reached either the "production" or the "mature" stages.[162] 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 56 percent[163] 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.[164]

Only tentative conclusions can be drawn from these numbers. It seems clear, however, that GPL is by far the most popular license among currently active open-source projects. It also seems clear that end users are not the main audience of open-source developers since developers or system administrators were more often the intended audience.

E. *Summary*

Open-source software has succeeded in only some areas. 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, so they can charge little more than their distribution costs for their distribution

161. Overall, however, the FSF prefers that developers use the GPL for their programs. *See* Stallman, *supra* note 96.

162. 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.

163. The exact number is 14,183.

164. The exact number is 19,164.

services; "buying" more software from them is likely to have little or no effect on the supply of programmers to develop open-source software.

PART 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. Section A examines the advantages and disadvantages of both approaches. Section B examines an oft-made claim that the open-source approach results in more innovation. Section C considers the evolution of open-source and proprietary software under government neutrality.

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, and GPL software to some extent, 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, society benefits most from an already-created piece of intellectual property when it is made available to all for free. Open source more or less does this, with the GPL being less attractive than other open-source licenses to commercial firms in this regard. Using code licensed under the GPL can impose restrictions that using code available under other licenses do not; namely, 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 and provide those fixes to other users. These advantages will appeal more to business users than to typical home users, of course, since medium and large businesses are likely to have technically adept staff to maintain their networks and corporate software.

Since technically adept users can inspect the source code if they so desire, it is possible that they might be able to create bug fixes more quickly than occurs 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 proprietary software. With open source, it would be difficult for a programmer to include code that would "spy" on unsuspecting users because other programmers could simply remove such code. Whether this theoretical advantage is a real-world advantage is not clear, since there is little evidence that commercial software engages in such behavior.[165]

b. Disadvantages

Because the source code is open to all, open-source developers have limited opportunities to earn a pecuniary return on the time and effort invested in their work. As discussed above, non-pecuniary rewards can certainly provide some motivation, but they do not appear any 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 often costly consumer research into usability and consumer needs that proprietary developers do in order to market to mass-market consumers; they have no easy way to earn a return on any such investments. As the SourceForge data suggest,[166] open-source projects are more often aimed at technical users, which sidesteps the usability issues. Similarly, the development process provides few incentives to identify and eliminate issues that might be problems for the less technically inclined. A government mandate that government employees (or citizens) use open source will force people to use software that is, on average, hard for them to use.

Open source is also subject to "fragmentation" (i.e. 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

165. *See* Part VI.C.1.b for a discussion on security and privacy concerns related to open-source software.

166. *Supra* notes 163 and 164.

another.[167] Linux has not yet fragmented to the same extent as Unix, and perhaps it never will, but Linux distribution differences have posed problems to developers and non-technical users.[168] Additionally, to the extent that open-source users take advantage of one of the claimed benefits of open source, their freedom to modify and customize code, fragmentation occurs.

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 initial investment. For example, one of the big movies of 2001 was the first installment of *Lord Of The Rings*. The movie's creators would have no incentive to devote as much time, effort, and money into the script, stars, and special effects if everyone who saw the movie could legally distribute copies to everyone else. Software is no different in this regard. The extent of intellectual property protection that computer software should receive might be debatable, but it is untenable, from an economic perspective, to argue that no protection should be available.

With proprietary software, a firm can control the destiny of its products. With this control, the firm 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.

The issue of fragmentation is related. One strength of both Windows and the Macintosh is that they provide consistent platforms on which applications can be run.[169] Developers for Windows (or the Macintosh) know that if they write their programs in certain ways, these programs should run on all computers meeting specific hardware and operating system requirements (e.g., Windows ME, Windows 2000, or later; OS X

167. The Linux Standard Base initiative is trying to overcome this particular problem. Linux Standard Base, *Standardizing the Penguin*, *at* http://www.linuxbase.org/ (last visited Jan. 23, 2003).

168. 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?*, *at* http://www.salon.com/tech/feature/1999/07/14/redhat/index1.html (Jul. 14, 1999); Charles Babcock, *Linux vs. Linux*, INTER@CTIVE WEEK, Mar. 19, 2000 (on file with MTTLR).

169. This consistency is important in providing the benefits of network effects to users.

or later). This is not true for Linux developers because of the free-for-all customization possibilities. A developer who wants to write a program that permits the linking of objects (such as putting a graph from a spreadsheet into a word processing document) can readily do so on either Windows or the Macintosh using standard methods for each; someone wanting to do so for Linux faces an uphill battle.[170]

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 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, in order to keep customers happy). Similarly, a technically adept user cannot customize a proprietary program except in the 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.[171] 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 under BSD-style licenses seem to have been truly innovative. Some of these early efforts are BIND, the BSD family of operating

170. 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*, CNETNEWS.COM, Feb. 26, 2001, *at* http://news.com.com/2102-1082-253153.html (last visited Jan. 23, 2003). 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 16, 2002, *at* http://zdnet.com.com/2102-1104-916331.html (last visited May 18, 2003).

171. *See, e.g.*, Marco Iansiti & Josh Lerner, *Evidence Regarding Microsoft and Innovation*, AEI-BROOKINGS RELATED PUBLICATION, Apr. 2002, *available at* http://aei.brookings.org/admin/pdffiles/innovation.pdf.

systems,[172] 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) and many modern projects seem more derivative, less innovative, and sometimes downright imitative. One of FSF's original objectives was to imitate Unix to develop a "free" version. The FSF work began with development tools, which was also an imitative step. 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 do not clone the user interfaces of either Windows or the Macintosh, but they attempt to bring the same kind of usability to Unix-like operating systems. 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 a standard set of capabilities (called Structured Query Language, or SQL).

Some observers have suggested that the open-source method tends to promote innovation more than the proprietary method and 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.[173]

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

> [Miguel] 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.[174]

172. See discussion *supra* Part IV.D.1.b.

173. O'Reilly Press Room, Open Source Pioneers Meet in Historic Summit (Apr. 18, 1998), *at* http://press.oreilly.com/pub/pr/796.

174. Russell Pavlicek, *Get Mono from .Net?*, INFOWORLD, Mar. 29, 2002, *at* http://archive.infoworld.com/articles/op/xml/02/04/01/020401opsource.xml.

In addition, Lawrence Lessig has argued that open source contributes to an intellectual "commons," which can serve as a springboard toward further development.[175] 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), BIND (domain name software), and Sendmail (an e-mail server).[176] Professor Lessig also said:

> 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.[177]
>
> 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.[178]

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.[179] And it is true that the Internet grew up around some of the products that Professor Lessig considers its "soul," particularly BIND and Sendmail. But what was the direction of causality? The Internet was not made commercially available until 1991 when the U.S. National Science Foundation removed restrictions on commercial use of NSFNET.[180] It was originally developed under government sponsorship to connect 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 usually written by its own technically adept users rather 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.

175. This view is not unlike the FSF's view of a GPL "club," mentioned above. "This free code builds a commons." LESSIG, *supra* note 7, at 57.

176. *Id.* at 56.

177. *Id.*

178. *Id.* at 57. Lessig also quotes similar sentiments from Alan Cox, "second only to Linus Torvalds in the Linux chain." *Id.*

179. A "protocol" can be thought of as a definition of how to do something. An "open" protocol is one that can be used freely by anyone; the code to implement an open protocol can be implemented in either open-source or proprietary software.

180. National Science Foundation, *The Internet*, *at* http://www.nsf.gov/od/lpa/nsf50/nsfoutreach/htm/n50_z2/pages_z3/28_pg.htm#answer3 (last visited May 18, 2003).

Of the five "soul" projects, Linux was the only one released under the GPL. The other products are all available under less commercially restrictive licenses.[181] How "innovative" were these projects? As economists, we do not pretend to provide firm answers to these questions. 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.[182] Our understanding is that BIND helped make the Internet possible,[183] and Sendmail ushered in the switch from older Internet e-mail to more modern Internet e-mail.[184] Perl was not the first scripting language,[185] but it does seem to have gained rapid acceptance and is widely used for some purposes.[186] Although Apache was neither the first nor only[187] Web server, it is directly descended from "patches" (hence the name) to the early NCSA Web server, whose development had stalled.[188] Thus, on balance, Apache's heritage and wide popularity suggest it should be considered innovative. These four products, all available under BSD-style licenses, have stronger claims to being innovative than does Linux.[189]

Of course, pointing out a few products that may be innovative, and contrasting with a few that may not, provides little to support the contention that open-source software was not innovative. What should be considered innovative? This is, even in principle, a difficult question to

181. 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 (last updated Jan. 6, 2003).

182. 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* NASA, *infra* note 284, and Sandia Lab News, *infra* note 285.

183. "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, *supra* note 150.

184. Red Hat Sendmail HOWTO, *supra* note 150.

185. Perl explicitly acknowledges "ancestors" like C, awk and sh. Elaine Ashton, *The Timeline of Perl and its Culture*, *at* http://history.perl.org/PerlTimeline.html (last visited May 18, 2003).

186. *See supra* text accompanying note 154.

187. Web servers from Microsoft, Sun, and other companies are also widely used.

188. Apache, *Apache HTTP Server Project*, *at* http://httpd.apache.org/ABOUT_APACHE.html (last visited May 18, 2003).

189. *See supra* Part IV.D.1.a for a discussion on Linux. *See supra* Part IV.D.1.b for a discussion on BIND, Sendmail, Perl, and Apache.

answer. Consider the following questions in the category of 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?
- Was the development of graphical word processors (true "what you see is what you get") innovative?

The breadth and depth of currently available commercial software came, in general, from investments made in pursuit of profits. Many products have copied their competitors and improved their features. 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 forbearers (if any) 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, as even a cursory glance at the projects hosted on SourceForge reveals.

C. *The Future Evolution of Open Source without Government Favoritism*

Absent government favoritism for (or against) open-source software, software users will choose software that best suits their needs, taking into account price, quality, ease of use, support, and other characteristics that they consider important. 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.[190] 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.

190. *See* EVANS, *supra* note 65; LIEBOWITZ, *supra* note 65.

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 that interest mass consumers, who value ease of use far more than do the technically adept. Open-source products, however, will certainly continue to provide competitive pressure to proprietary software developers. This pressure will likely persist because the open-source production method is geared towards serving the technically adept and not the mass market.[191] Even some major proponents of open-source software take that view.[192]

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,[193] but today, the integrated server vendors and integrated workstation vendors continue to ship their own Unix flavors: Solaris, AIX, HP-UX, Irix, Tru64 Unix, and so forth. Some vendors, most notably IBM, are now interested in Linux.[194] Others are less so. For example, Sun uses Linux on low-end servers based on Intel-compatible hardware,[195] but it uses only its own version of Unix on all computers based on its own SPARC processor family.[196]

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;

191. 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.

192. *See*, *e.g.*, Brian Behlendorf, *Open Source as a Business Strategy*, *in* OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION 149 (Chris DiBono et al., eds. 1999), *available at* http://www.oreilly.com/catalog/opensources/book/brian.html.

193. The Open Group: History and Timeline, *UNIX Past*, *at* http://www.unix-systems.org/what_is_unix/history_timeline.html (Jun. 25, 2001).

194. Evans, *supra* note 130.

195. Sun Store U.S., *Sun Fire B100s Blade Server*, http://store.sun.com/catalog/doc/BrowsePage.jhtml?catid=93754 (last visited May 6, 2003).

196. Sun Store U.S., *Sun LX50 Server*, http://store.sun.com/catalog/doc/BrowsePage.jhtml?cid=85662&parentId=48589 (last visited May 6, 2003).

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, the scope of that potential is unknown. Moreover, Linux faces its own potential for fragmenting, as discussed above.

PART 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.[197] State and local governments spent another $4.5 billion on prepackaged software in 2000.[198] Like profit-maximizing 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 with regard to price, technical advantages, or both. 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, the Bundestag in Germany recently decided, based in part on a study it commissioned, to use Linux on most servers while using Windows for clients on desktops.[199] While one could argue whether this is the right decision and insinuate that the Bundestag had other motives, it is indistinguishable 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.[200] But, support has also come

197. *2001 Annual NIPA Revision*, Bureau of Economic Analysis, and the U.S. Department of Commerce, Tables 1, 11 (Aug. 2001), *available at* http://www.bea.doc.gov/bea/papers/tables.pdf (last visited May 18, 2003).

198. *Id.*

199. *Ältestenrat Stimmt für Linux auf Bundestags-Servern* [The Ältestenrat, council of the oldest members, votes for Linux for the Bundestag servers], HEISE ONLINE, Mar. 14, 2002 [hereinafter Ältestenrat Stimmt], *available at* http://www.heise.de/newsticker/data/anw-14.03.02-012/.

200. Stallman Addresses Brazilian Congress, *supra* note 6.

from outside the open-source community. For example, as will be discussed infra, Professor Lessig argued that governments should support open source.[201] Governments have established study groups to consider government support for open source, and politicians in many countries have introduced legislation to help open source. Few governments, to date, have enacted explicit preferences for open-source software; the most prominent are a handful of Brazilian cities.

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?

Section 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. Section 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. Section 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. Section D evaluates from an economic standpoint a particular government policy that would require 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 which is measured, roughly speaking, as the value that society gets from its scarce resources. There is a body of theoretical literature,

201. LESSIG, *supra* note 7, at 247.

starting with Adam Smith's *Wealth of Nations*,[202] 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

> [L]ed 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.[203]

Modern mathematical models explain how this decentralized process maximizes the collective good in formal terms.[204]

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 their alternatives at maximizing social welfare. Much of the 20th century was devoted to the grand experiment to see 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 led to great dissatisfaction. Beginning in the mid-1970s, this dissatisfaction spawned efforts to greatly reduce government involvement in industry. The United States and Great Britain led 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 necessary for an intervention to make the public economically better off. First, identify a market failure—provide an explanation why the market, presumed to be efficient most of the time, does not work. Second, identify a government solution that is likely to

202. ADAM SMITH, AN INQUIRY INTO THE NATURE AND CAUSES OF THE WEALTH OF NATIONS 423 (1937).

203. *Id.* at 423.

204. Kenneth J. Arrow and John R. Hicks shared the 1972 Nobel prize in economics for early work in this area. Nobel *e*-museum, *Bank of Sweden Prize in Economic Sciences in Memory of Alfred Nobel 1972, at* http://www.nobel.se/economics/laureates/1972/ (last modified Jun. 16, 2000).

correct the market failure at a reasonable cost, 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 Akerlof, Joseph Stiglitz, and Michael Spence—who identified a raft of problems over the years.[205] Examples of situations in which market forces may not maximize public welfare include the following, all long known to economists and governments:

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 could two; 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.

205. The Royal Swedish Academy of Sciences Press Release, The 2001 Sveriges Riksbank (Bank of Sweden) Prize in Economic Sciences in Memory of Alfred Nobel (Oct. 10, 2001), *at* http://www.nobel.se/economics/laureates/2001/press.html.

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,[206] 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 many different market failures that cost the public more than the original failure. 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 they are needed to remedy a market failure. So, the fourth problem results from erroneously and expensively fixing a market failure that may not have existed in the first place.

Economists have widely acknowledged that government cures may be worse than the disease. For example, Professor Stiglitz wrote:

> Whenever 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.[207]

Professors Michael Katz and Harvey Rosen said:

> 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 cases 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

206. Ronald H. Coase, *The Problem of Social Cost*, 3 J.L. & ECON. 1 (1960).
207. STIGLITZ, *supra* note 9, at 163.

> theory it 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. (emphasis in original)[208]

As with most real-world markets, software markets do not work as perfectly as a benevolent central planner would like. To begin with, most economic work demonstrating 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, with thousands of producers competing with each other, full disclosure of information on prices and quality, and a more or less static environment. In contrast, the software industry involves substantial fixed costs, the creation of intellectual property, and a substantial gap between prices and marginal costs for successful products.[209] Plus, the software market is dynamic and subject to rapid rates of technological change, with leapfrogging competition between products.[210] As noted above, however, the fact that competition in software does not resemble that in wheat does not necessarily mean that the government should step in and regulate the software industry.

Economists have only begun to analyze formally the properties of the kind of competition seen in software, and the work has progressed slowly because the problem is actually quite difficult.[211] To date, only a

208. MICHAEL L. KATZ & HARVEY S. ROSEN, MICROECONOMICS 399 (Irwin/McGraw-Hill 1998).

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

210. Elzinga, *supra* note 44; David S. Evans & Richard Schmalensee, *Some Economic Aspects of Antitrust Analysis in Dynamically Competitive Industries*, *in* INNOVATION POLICY AND THE ECONOMY 1 (Adam B. Jaffe et al. eds., 2002).

211. 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 L. Katz & Carl Shapiro, *Network Externalities, Competition, and Compatibility*, 75 AM. ECON. REV. 424 (1985); Michael L. Katz & Carl Shapiro, *R&D Rivalry with Licensing or Imitation*, 77 AMER. ECON. REV. 402 (1987); Gene Grossman & Carl Shapiro, *Dynamic R&D Competition*, 97 ECON. J. 372 (1987); Drew Fudenberg & Jean Tirole, *Pricing a Network Good to Deter Entry*, 48 J. INDUS. ECON. 373 (2000); Jennifer F. Reinganum, *The Timing of Innovation: Research, Development, and Diffusion*, 1 HANDBOOK OF INDUSTRIAL ORGANIZATION 849 (Richard Schmalensee & Robert D. Willig, eds. 1989); JEAN TIROLE, THE THEORY OF INDUSTRIAL ORGANIZATION 389–421 (1988); PAUL STONEMAN, ECONOMIC ANALYSIS OF TECHNOLOGICAL CHANGE (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 341, 341–413 (2001).

very few papers have attempted theoretical investigations of competition between open-source and proprietary software. In general, these papers reach conclusions that are consistent with the view that banning the use of proprietary software (for all consumers or even just for government consumers) would make consumers generally worse off.[212]

Further, as discussed above, the history of the software industry suggests that the industry has worked quite well from the standpoint of consumers without government mandates concerning software. We saw earlier that output has increased quickly, quality-adjusted prices have fallen, and innovation has been rapid. All of these made consumers better off. Although firms have dominated particular categories of software, at least for a time, the industry is fairly unconcentrated and participants generally consider it highly competitive.[213] These features, too, strongly suggest that the industry operates to the benefit of consumers. There is a considerable amount of entry into and exit from the industry, and key players turn over with some frequency. Economists take these characteristics as showing the competitive health of an industry.[214] Of course, there could be problems that need fixing and solutions worth considering, but any such problems would need to be identified and solutions targeted at those specific problems—not at the industry as a whole.

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

212. James Bessen, *Open Source Software: Free Provision of Complex Public Goods*, *at* http://www.idei.asso.fr/Commun/Conferences/Internet/OSS2002/Papiers/Bessen.PDF (working version Jun. 2002); Gilles Saint-Paul, *Growth Effects of Non Proprietary Innovation*, *at* http://www.idei.asso.fr/Commun/Articles/St-Paul/os1.pdf (Oct. 16, 2001); Klaus Schmidt & Monika Schnitzer, *Public Subsidies for Open Source? Some Economic Policy Issues of the Software Market*, http://www.idei.asso.fr/Commun/Conferences/Internet/OSS2002/Papiers/Schmidt.pdf (preliminary version Jun. 14, 2002).

213. *See*, *e.g.*, Elzinga, *supra* note 44. *See also* Microsoft Corp., Form 10-K for Fiscal Year Ended Jun. 30, 2001 (citing "the software business is intensely competitive and subject to rapid technological change"); Oracle Corp., Form 10-K for Fiscal Year Ended May 31, 2001 (stating "the computer software industry is intensely competitive and rapidly evolving"); Sun Microsystems, Form 10-K for Fiscal Year Ended Jun. 30, 2001 (noting "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 Dec. 31, 2001 (discussing that "the software and Internet industry is intensely competitive").

214. DENNIS W. CARLTON & JEFFREY M. PERLOFF, MODERN INDUSTRIAL ORGANIZATION 56, 56–58 (3d ed. 2000).

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. In general, these decisions are no different than those that must be made by countless private firms and individuals around the world. There are some differences, however. When legislators get involved, these decisions are 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; for instance, private firms and individuals make their decisions without commission recommendations. As a result, special commissions, legislative proposals, and the like demonstrate that government decisions may be made on grounds that a private company would not consider (e.g., a private company's goal is to make money, not to please the voting public).

1. Initiatives

a. European Commission and European Parliament

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 concluded 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."[215] The study also argued:

215. Patrice-Emmanuel Schmitz, *Part 3: The Open Source Market Structure* 7, *in* STUDY INTO THE USE OF OPEN SOURCE SOFTWARE IN THE PUBLIC SECTOR, Interchange of Data be-

"the requirement of the use of [open source software (OSS)] should be justified under Article 81, paragraph 3" of the European Community ("EC") treaty;[216] "software patents present a major threat concerning a fundamental liberty";[217] "OSS is considered to better respect standards";[218] "OSS in general and GPL in particular permits a greater rate of innovation, with greater efficiency";[219] and "with OSS you will have no (or less) backdoor(s), no electronic spy that may be totally hidden somewhere in the software."[220] Thus far, although the Commission appears to support open source, it has not acted on these findings, which appear to have both economic and ideological elements (e.g., "fundamental liberty" has little to do with economics).

To take another example of support, the European Parliament adopted a Resolution on September 5, 2001 that "urge[d] 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 asked "the Commission to lay down a standard for the level of security of e-mail software packages, and to place those packages with non-public source code in the 'least reliable' category."[221] The Parliament's recommendations are only that and were not adopted by the Commission as far as we know.

tween Administrations, Jun. 2001, http://europa.eu.int/ISPO/ida/export/files/en/835.pdf (last visited May 18, 2003).

216. *Id.* 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."

217. *Id.* at 74.

218. *Id.* at 15.

219. *Id.* at 39.

220. *Id.* at 16.

221. European Parliament resolution, *supra* note 2; *see also* Paul Meller, *European Parliament Adopts Echelon Report*, COMPUTERWORLD, Sept. 5, 2001, http://www.computerworld.com/securitytopics/security/story/0,10801,63553,00.html.

b. Germany

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 information technology ("IT") environment. It decided to follow the luK (eleven-member committee comprised of representatives from major political parties) recommendation[222] to install Linux on approximately 150 servers and Windows XP on 5,000 desktops.[223] As noted earlier, the decision does not necessarily amount to a government intervention in open source since governments, just like businesses, must make IT decisions. However, numerous political statements in favor of open source accompanied the debate in the Bundestag over the migration issue.[224] 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 listed the claimed advantages of open source: stability, better potential to be tailored to users' needs, and high security.

222. The luK recommendation was based on the INFORA study commissioned by the Ministry of Interior, which analyzed various scenarios of Bundestag IT environment.

223. Ältestenrat Stimmt, *supra* note 199. *See also Linux für die Server, Windows für PC im Abeorgnetenhaus* [Linux for the servers, Windows for PC in the parliament], *at* http://www.heute.t-online.de/ZDFheute/artikel/0,1251,COMP-0-178460,00.html (Feb. 2, 2002); *Studie empfiehlt Schily Server-Umrüstung auf Linux* [German Minister of the Interior added to change servers to Linux], HEISE ONLINE, Dec. 7, 2001, http://www.heise.de/bin/nt.print/newsticker/data/jk-07.12.01-008/?id=80204dba&todo=print.

224. For instance, a Social Democrat, Jörg Tauss, announced, "[M]y wish would be to declare the entire Bundestag a Microsoft-free zone." See John Ness & Stefan Theil, *The Threat of a Linux Generation,* NEWSWEEK INT'L, Mar. 11, 2002, *available at* http://www.my-opensource.org/lists/myoss/2002-03/msg00010.html. 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 (on file with author). 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. *Il parlamento tedesco dice sì a Linux*, WEBNEWS, Mar. 11, 2002, *at* http://webnews.html.it/focus/181.htm.

The resolution called on the government to introduce open source in the federal administration and stated open source should be used wherever it would lead to cost savings. Again, any legislative intervention into an administrative issue reflects an unwarranted 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.

c. France

The French public sector has widely adopted open source solutions since the beginning of 1999 and continues to move in the direction of complete open-source infrastructure.[225] In fact, several governmental institutions have already switched to open-source software. 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.[226] Other agencies moving to open source include the Ministry of Justice, the Department of National Education, and the Ministry of Economy, Finance and Industry.[227]

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 some French views on open source. One parliamentary bill submitted in December 1999 would require that all software used by the government be open code and would create an open source agency to oversee the transition process.[228] 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;

225. Schmitz, *supra* note 215, at 28.

226. *See* Mandrake Linux, *Le Ministère de la Culture, connu pour son engagement en faveur du logiciel libre, confie à MandrakeSoft la mise en place de serveurs Linux dans six musées de province*, Oct. 16, 2000, http://www.linux-mandrake.com/fr/pr-culture.php3.

227. Schmitz, *supra* note 215, at 33–35.

228. Proposal for a Bill Purporting to Generalize the Use of the Internet and of Free Software in the Administration (1999) (Fr.), *available at* http://www.senat.fr (on file with author).

and 4) principles for compatibility, competition in software, respect for privacy, and civil liberties should be established.[229]

d. Brazil

A few Brazilian locales have succeeded in requiring the government to use open source.[230] In June 2001, the City Council of Amparo, a city in São Paulo state, passed a law requiring that the municipal government prefer free, unrestricted open-source software.[231] The municipality of Recife also has an open source preference pursuant to an executive decree in April 2000.[232] In December 2001, the Chamber of Councilmen in Porto Alegre approved a project that sets the conditions for open source use in the municipal administration.[233] Several other Brazilian cities and states have considered or are considering open source preference proposals.[234]

e. Italy and Spain

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."[235] 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

229. Proposal for a Parliamentary Bill Purporting to Reinforce Freedom and Security of Consumers and to Enhance Competition in the Information Society (2000) (Fr.), *available at* http://www.assemblee-nat.fr/propositions/pion2437.asp (last visited May 18, 2003).

230. *See* Paul Festa, *Governments push open-source software*, CNET NEWS.COM, Aug. 29, 2001, *at* http://news.com.com/2100-1001-272299.html?legacy=cnet.

231. *See* Linux On*, Amparo é a primeira cidade de São Paulo a adotar Software Livre* [Amparo is the first city in the State of São Paulo to adopt open-source software] (Sept. 3, 2002) (on file with author).

232. *See* Cláudio M. Machado, *Recife na rota do Software Livre* [Recife in the route of the open-source software], *at* http://www.pernambuco.com/tecnologia/arquivo/softlivre1.html (last visited May 8, 2003).

233. *See* Projecto Software Livre—RS, *Aprovado Projecto de Lei Sobre Software Livre em Porto Alegre* [Approved act about open-source software in Porto Alegre] (Dec. 14, 2001) (on file with author).

234. 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. *Id.*; *see also* Festa, *supra* note 230.

235. Festa, *supra* note 230.

public administration of the Province of Pescara.[236] In Spain, the Canary Islands Regional Parliament approved a proposal to urge the regional government, in partnership with local authorities and companies, to promote use of OSS through training courses and increasing public awareness about OSS availability.[237]

f. Venezuela

The Venezuelan government announced "all software developed for the government must be licensed under the GPL," with the general objective being "open source whenever possible, proprietary software only when necessary."[238] The government's open source mandate seems motivated by a desire to help local programmers. "The government and the people of Venezuela were increasingly concerned that over 75 percent of the funds for software licenses went to foreign nations, 20 percent to foreign support agencies, and only 5 percent to Venezuelan programmers."[239] The government's specification of the use of the GPL seems politically motivated, based in part on the "great wealth of technical advisors"[240] who are active Linux developers and users and who "were clearly influential in reaching this new policy decision."[241]

g. China, Singapore, and Peru

A software development group set up by the Chinese government has demonstrated progress in developing its own Linux version to replace Windows and Unix on all government servers and PCs.[242] "The group's goal is to develop an entire desktop environment with open source technology for the government,"[243] including office productivity software.

236. *See Open Source Nella Provincia di Pescara*, *at* http://www.interlex.it/pa/ppescara.htm (May 15, 2002).

237. *See Boletin Oficial del Parlamento de Canarias*, http://www.parcan.es/pub/Bop/5L/2001/166/bo166.pdf (Jul. 20, 2001).

238. Brian Proffitt, *Venezuela's Government Shifts to Open Source Software*, LINUXTODAY, Aug. 30, 2002, *at* http://linuxtoday.com/mailprint.php3?action=pv&ltsn=2002-08-30-011-26-NW-LL-PB.

239. *Id.*

240. *Id.*

241. *Id.*

242. Matt Berger, *LinuxWorld Expo: Chinese Government Raises Linux Sail*, INFOWORLD, Aug. 13, 2002, *at* http://archive.infoworld.com/articles/hn/xml/02/08/13/020813hnchina.xml.

243. *Id.*

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 requiring the use of free software in government computers is currently in the legislature.[244] Similar bills have been proposed elsewhere in Latin America (e.g., Argentina).[245]

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, instead mentioning issues such as independence, innovation, competition, and helping domestic industries. 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.

244. Julia Scheeres, *Peru Discovers Machu Penguin*, WIRED NEWS, Apr. 22, 2002, *at* http://www.wired.com/news/business/0,1367,51902,00.html; Thomas C. Greene, *MS in Peruvian Open-Source Nightmare*, THE REGISTER, May 5, 2002, http://www.theregister.co.uk/content/4/25157.html.

245. 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* Marcelo Dragan, *Bill 5613-D-00*, http://www.proposicion.org.ar/proyecto/leyes/5613-D-00/index.html (last modified Jan. 7, 2003).

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 stable than proprietary software. For example, the 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.[246] 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 (e.g., against viruses).[247] 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.[248] A related concern is that "some versions of Windows contain backdoors designed to grant the U.S. National Security Agency access to users' data"[249] which would compromise users' privacy.

b. Cost Savings

Some government-sponsored studies have claimed that using open source saves money. German experts said the use of open-source software in public administration would save the federal government €130 million and €2.6 billion countrywide.[250] 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.[251]

c. Independence

On various occasions, the German government has expressed its concerns about the administration's dependency on single software

246. *See Zusammentassung der Rede der Staatssekretärin Brigitte Zypries* [Resume of the Speech of Secretary of State Brigitte Zypries], http://linux.kbst.bund.de/auftakt/vortraege/zypries/ (Jun. 8, 2000).

247. *See id.*

248. *See Open Source Software im Parlament* [Open-source software in Parliament], http://linux.kbst.bund.de/bundestag/bt-pp14.199.html (last modified Nov. 16, 2001).

249. Rick Perera, *German Parliament Considers Linux Switch*, ITWORLD.COM, Oct. 16, 2001, *at* http://www.itworld.com/Comp/2384/IDG011016germanlinux/pfindex.html.

250. *See Ein kühner Plan: Linux statt Windows* [A Bold Plan: Linux Instead of Windows], Oct. 7, 2001, http://teamx.cc/news_2002_2.htm.

251. *See Bundesregierung gegen "Monokultur" bei Software* [Federal government against monoculture with respect to software], HEISE ONLINE, Jul. 5, 2001, http://www.heise.de/newsticker/data/odi-05.07.01-000/.

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 dependence on single software providers by adopting open source.[252] At the regional level, Mrs. Harms, a Green Party Member of the Lower Saxony regional Parliament, stated that a Linux platform complemented with open-source and commercial products should relieve the dependence on a single provider.[253]

d. Innovation

A number of the proposals to assist open source have claimed that open source is more innovative than 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."[254] 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,"[255] 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."[256] On a national level, in its government-commissioned study on patent protection of software products, the Fraunhofer Institute and Max-Planck-Institute stated "[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."[257] Professor Lessig, whose views we discuss both above and

252. *See* Federal Ministry of Interior Affairs, *Linux-Tag: Abhängigkeit von Softwareherstellern verringern* [Linux-day: Reduce Dependency on Software Producers], DIE WELT, Jul. 6, 2001, http://www.bmi.bund.de/top/dokumente/Rede/ix_47733.htm.

253. *See* Lower Saxony Parliament, *Niedarsächsischer Landtag—14. Wahlperiode, Drucksache 14/1492* at 1 (Aug. 2, 2000) (reporting a brief question by Deputy Frau Harms), http://www.landtag-niedersachsen.de/Drucksachen/Drucksachen_14_2500/1001-1500/14-1492.pdf.

254. Schmitz, *supra* note 215.

255. Bud P. Bruegger, et al., *SPIRIT—Accelerating the Uptake of Open Source in Healthcare*, *at* http://www.sistema.it/LinuxAfrica2001/ (last visited May 9, 2003).

256. *Id.*

257. Knut Blind et al., *Micro- and Macroeconomic Implications of the Patentability of Software Innovations. Intellectual Property Rights in Information Technologies between Competition and Innovation* 6, http://www.sicherheit-im-internet.de/download/ softwarepatentstudie_e.pdf (Sept. 2001).

below, has also based his support on the proposition that open-source software has been innovative.[258]

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, noted that open-source software played an important role in bringing competition to the software market.[259] 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.[260] The Commission's 2001 IDA study also noted the proposition that open source would help competition. The study 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."[261]

f. Helping Domestic Industries and Other Nationalistic Motives

The German government's national interest has yet another argument in favor of open source. As Professor Lutterbeck stated 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."[262] He concluded that the open-source area not yet dominated by the United States is of great economic importance

258. *See* LESSIG, *supra* note 7, at 56–57.

259. *See Bundesregierung gegen "Monokultur" bei Software,* HEISE ONLINE, Jul. 5, 2001, *at* http://www.heise.de/newsticker/data/odi-05.07.01-000/.

260. Beratende Stellungnahme des Landesrechnungshofs Schleswig- Holstein zur Informationstechnik der Landtagsverwaltung [Audit Study of the State Court of Audits Schleswig-Holstein on the Information Technology of the State Parliament Administration], Doc. No. LRH Pr 1255/2000, at 13, Oct. 31, 2000 [hereinafter Beratende], http://www.lrh.schleswig-holstein.de; Deliberation of the Issue in the Budget Committee of the Regional Parliament 11–13, Jan. 18, 2001, http://www.sh-landtag.de/infothek/wahl15/aussch/finanz/niederschrift/2001/15-033_01-01.pdf.

261. Schmitz, *supra* note 215, at 68.

262. Bernd Lutterbeck et al., *Security in Information Technology and Patent Protection for Software Products: a Contradiction?* 4 (2000), http://www.sicherheit-im-internet.de/download/BMWi_Gutachten_englisch.pdf (last visited May 18, 2003).

for Germany.[263] 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 considered a special opportunity for the European software industry and should not be missed.[264]

Helping domestic programmers was also explicitly stated as "one of the big reasons" for the GPL mandate in Venezuela.[265]

g. Ideological

Most recent open-source software seems to be developed under the GPL. This is due to FSF's influence, whose motivation in developing and supporting the GPL is purely ideological. Some of the justifications provided for government policies in support of 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."[266]

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."[267]

Politics was one of the reasons for the Venezuelan government's recent mandate for governments to use GPL software. The government was urged by an advisory group that included Linux developers and users.[268] If the government had been interested solely in helping domestic programmers, it could have mandated that the software in question either be in the public domain or have any one of several licenses. For example, in addition to allowing the GPL, the mandate might have accepted

263. *Id.*

264. Antrag [Proposed Legislation] under Deutscher Bundestag Drucksache 14/5246 at 5, Jul. 7, 2001, http://dip.bundestag.de/btd/14/052/1405246.pdf; Deutscher Bundestag Plenarprotokoll 14/199 [Federal Parliament Minutes 14/199] at 19577, Nov. 9, 2001 (approving proposed legislation), http://dip.bundestag.de/btp/14/14199.pdf.

265. Proffitt, *supra* note 238.

266. Stallman Addresses Brazilian Congress, *supra* note 6.

267. Working Group on Libre Software, *Free Software / Open Source: Information Society Opportunities for Europe? Version 1.2* (Apr. 2000), http://eu.conecta.it/paper.pdf (last visited May 18, 2003).

268. Proffitt, *supra* note 238.

software licensed under the LGPL, the BSD license, or other non-restrictive licenses—not just the ideology-based GPL.

C. *Economic Arguments for Helping Open Source*

Economic arguments for helping open source are based on the premise that open source is "better" in some ways and therefore the government should help promote it. In theory, two sorts of arguments might be made. One is that open source is superior and should be used more by prudent purchasers, and the government is no different from business. The other is that open source could provide various economic benefits if successful, so government should give it a boost.

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

> 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.[269]

His argument 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 IT department.[270] 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. Professor Lessig's passage, though, may be based on a market failure concept to which we return below.

The economic 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.

269. LESSIG, *supra* note 7, at 247.

270. *See, e.g.* Lawrence Lessig, *Code and Commons*, Keynote at the Conference on Media Convergence (Feb. 9, 1999) (transcript *available at* http://cyberlaw.stanford.edu/lessig/content/articles/works/fordham.pdf; Lawrence Lessig, *May the Source Be with You*, WIRED, Dec. 2001, http://www.wired.com/wired/archive/9.12/lessig_pr.html.

We begin with the premises and then turn to the claims that government assistance is needed.

1. Claims about the Superiority of Open-Source Software

a. Innovation

As we discussed above, there is no basis for claiming that 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, which generally have been released under the GPL.[271] 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. A general claim, however, that open source is more innovative that proprietary software cannot be made.

b. Security and Privacy Concerns

Claims by its 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.[272] 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.[273]

271. *See, e.g.,* Richard Stallman, *UNESCO and Free Software*, UNESCO Free Software Portal, *at* http://www.unesco.org/webworld/portal_freesoft/stallman_011001.shtml (last updated June 26, 2001). This article addresses UNESCO's support of Free Software Foundation that promotes GNU/Linux system[s].

272. Robert Lemos, *Too Much Trust in Open Source?,* ZDNET NEWS, Mar. 20, 2002, *at* http://zdnet.com.com/2100-1104-864256.html; Wayne Rash, *Proprietary Apps Have Security Problems—But So Does Open Source*, ZDNet News, Apr. 4, 2002, *at* http://zdnet.com.com/techupdate/stories/main/0,14179,2857736,00.html.

273. *See, e.g.,* Mitch Wagner, *Virus Attacks Linux, Windows Systems*, INTERNETWEEK.COM, Mar. 30, 2001, *at* http://www.internetwk.com/story/INW20010330S0007.

Claims 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 and transmit usage data back to the software vendor for marketing or other purposes. In practice, open source cannot easily do this, 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 supposedly secure information; again, a skilled programmer could excise such code from an open-source program. Such claims are difficult to evaluate. It is certainly true that open source can generally avoid potential privacy and security problems like these. There is little evidence that proprietary programs actually engage in such activity.[274] Much of the argument seems based on speculation (what could happen) rather than fact (what does happen).[275] 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 software.

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 than on clients. This client-side preference for proprietary software strongly suggests that, despite the client-side cost savings (at least acquisition costs, if not training and support costs), open source generally has been inadequate in overcoming the technical advantages of proprietary software. Where formal studies have been conducted, the results have not always been conclusive. 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,

274. 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 & Rachel Konrad, *PC Invaders*, CNET NEWS.COM, Apr. 18, 2002 *at* http://news.com.com/2009-1023-885144.html.

275. 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 38–39, 56, 60–64 (2001).

other costs such as training, setup, support and development should be taken into account. The study concluded that there was no clear economic winner.[276]

2. Arguments for Promoting Open Source

The less sophisticated argument for promoting open source is that it is "good" so let us have more. The more sophisticated argument is that government assistance for open source 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 government promotion of open-source use seem to be based on nothing more than the observation that open-source software has certain features that are claimed to be 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.")[277] 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.[278] To the extent that open-source software is better than proprietary software in meeting government IT 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 occasional attempt to legislate the use of open source implies that more is going on here than a simple technical debate.

276. Beratende, *supra* note 260; Deliberation of the Issue in the Budget Committee of the Regional Parliament, *supra* note 260.

277. Michel Sapin, Address at the Opening of the Second Meeting on Free Software in the Administration (Nov. 15, 2001), *available at* http://www.fonction-publique.gouv.fr/communications/discours_archives/discours-200111151520.htm.

278. *See* Zusammenfassung der Rede der Staatssekretärin Brigitte Zypries [Summary of the Speech of State Secretary Brigitte Zypries], Koordinierungs-und Beratungsstelle der Bundesregierung für Informationstechnik [Federal Agency for the Coordination and Advice on Information Technology], http://linux.kbst.bund.de/auftakt/vortraege/zypries/ (speech at the University of Applied Science [Fachhochschule] of the Federation, Brühl on Jun. 8, 2000).

We make 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. That open-source software has seen greater relative successive on servers than on clients strongly suggests that users make exactly those kinds of comparisons. Second, there is no basis for claiming that an advantage of any particular open-source program over a proprietary counterpart will persist. As we saw above, there is nothing intrinsic in open-source software that ensures that it will be more secure than proprietary software. Third, there seems to be a suggestion that because there is something "good" about open source (better security, it is free, etc.) 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 have bad track records at picking technology winners and losers.

The potential for "innovation" by open-source software is not, by itself, a sensible economic reason for favoring open source. A private firm deciding what mix of software will best meet its needs over a relevant time horizon would care about which of the specific products and technologies are most cost-effective over that time horizon; the firm would not care whether some production method tends over the long run to be more "innovative" than another.

b. Promoting Open Source to Increase Competition

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

Many businesses ensure that they have at least two suppliers. That provides several advantages. First, it provides a supply source if one vendor cannot perform. For example, a business that requires a critical component may not want to risk running out of that component so it will retain a second supplier as back up. An explanation like this can hardly apply to a software supplier, since capacity and manufacturing problems are not relevant for software.

Second, it ensures supply competition; the availability of a second supplier can help discipline the prices of the first. Indeed, it is theoretically possible that a company would be willing to pay a higher price or accept a lower-quality product just to get the benefits of second source. 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 economically significant entry barriers 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, due to competition for distribution of the product. This problem arises regardless of the motivation underlying the procurement preference.

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.[279] 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

279. *IDC Server Tracker* database, Q196-Q302, *at* http://www.idc.com (last visited Nov. 26, 2002).

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.[280] We do not see any basis other than merit for having the government choose Linux operating systems for its servers.

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.[281] 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—purchases enough client operating systems to have much effect on Linux's success; moreover, as discussed above, procurement policies seem particularly ill-suited for promoting the development of open source.
2. The open-source software method appears unlikely to serve consumer needs on the client side. Unlike server operating systems, 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 or incentives for designing software for the masses.
3. There is no apparent reason why Linux should be the "second source" alternative if there is to be one. The Macintosh OS or OS/2 could be better potential choices. There are already 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 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 knowledge of user interfaces, program capabilities, and the like can be readily transferable. Deliberately avoiding

280. *Id.*

281. Al Gillen et al. *IDC Report #25118, Worldwide Client and Server Operating Environments Market Forecast and Analysis Summary, 2001–2005*, Table 2, p. 12 (Aug. 2001).

> 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 general public's network effects 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 this 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.[282] Filling in his thoughts, the argument seems as follows: if the government promotes the Linux operating system, that would increase the share of computers running the Linux operating system; more applications would be written to run on the Linux operating system; there would be more competition for Windows; and consumers therefore would be better off. This argument could be made for many products with network effects that the government uses, from telecommunications systems to credit cards. Professor Lessig provides neither theory nor fact supporting government aid of 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 through universities and government research laboratories. In general, government sponsorship of R&D efforts, for other industries as well as software, is justified as overcoming market failures and providing beneficial externalities.

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 software research either went into the public domain, remained in the hands of the military, or was spun off for

282. LESSIG, *supra* note 7, at 247.

commercial purposes.[283] We make no claims about specific products but note that the Internet arose from U.S.-sponsored research. Whatever policies were in place at the birth of the Internet (which notably did not include sponsoring GPL software), they seem to have succeeded technologically.

In recent years, however, substantial government R&D support has been directed at GPL software. For example, NASA developed the original Beowulf clustering software for Linux, released under the GPL.[284] More advanced clustering software was developed at Sandia National Laboratories, also released under the GPL.[285] The next version of the Reiser File System is sponsored primarily by the Defense Advanced Research Projects Agency (DARPA) and will be licensed under the GPL.[286] And, in April 2002, Sandia National Laboratories released its "DAKOTA Toolkit" under the GPL.[287]

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.[288] If such policies are appropriate for other

283. 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* National Science Foundation, *supra* note 180.

284. *See* NASA at Goddard Space Flight Center, *Beowulf at NASA/GSFC*, *at* http://beowulf.gsfc.nasa.gov/ (last modified Apr. 24, 2000). *See also*, Sterling, *supra* note 5.

285. *Laboratories Support, Facilities, and Human Resources*, 54 Sandia Lab News (2002), *at* http://www.sandia.gov/LabNews/LN02-22-02/LA2002/la02/support_story.htm (last modified Feb. 28, 2002).

286. Reiser, *supra* note 128.

287. Sandia National Laboratories Press Release, Sandia's DAKOTA Toolkit on Web and Available for Free (Apr. 8, 2002), http://www.sandia.gov/media/NewsRel/NR2002/DAKOTA.htm.

288. 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 on Government Relations, *The Bayh-Dole Act: A Guide To The Law and Implementing Regulations* (Sept. 1999), *at* http://216.239.51.100/

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.[289]

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; in contrast, 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."

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 he then rejects them.[290] 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."[291] 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."[292]

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,

search?q=cache:PaG-7h5QdxEJ:www.cogr.edu/docs/Bayh_Dole.pdf+The+Bayh-Dole+Act:++A+Guide+To+The+Law+and+Implementing+Regulations+&hl=en&ie=UTF-8.

289. Indeed, the GPL is incompatible with any assertion of patent rights.

290. LESSIG, *supra* note 7, at 329–330.

291. *Id.*

292. *Id.*

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 patents in the special patent pool and would donate few, if any, patents to the pool. The U.S. government does not fund pharmaceutical research in ways that prevent the use of that research by firms making proprietary products. But funding R&D for GPL software has exactly that effect in software—preventing its use by firms making proprietary products.

In short, the economic justifications for government support of R&D seem no different for software than for other industries. The economic reasoning that leads the government to favor commercialization of R&D results in other industries applies equally well to software. If for some reason software is deemed to be different, then government policies should favor releasing R&D software research into the public domain or under BSD-style licenses, to permit their widespread use.

PART 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 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 more users. Open-source 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?"[293] Whether "open code" in any given situation is actually "as powerful" as "closed code" is an everyday 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 (e.g., network effects discussed above), we see no reason to believe that they are more important for "open code" 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. 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.

293. LESSIG, *supra* note 7, at 247.