1-1-2001

Allchin's Folly: Exploring Some Myths About Open Source Software

Joe Miller
University of Georgia School of Law, getmejoe@uga.edu

ALLCHIN'S FOLLY: EXPLODING SOME MYTHS ABOUT OPEN SOURCE SOFTWARE

JOSEPH SCOTT MILLER*

> "Open source is an intellectual-property destroyer," [Microsoft Group VP James] Allchin said. "I can't imagine something that could be worse than this for the software business and the intellectual-property business." . . . "I'm an American, I believe in the American way," he said. "I worry if the government encourages open source, and I don't think we've done enough education of policy makers to understand the threat."[1]

The twists and turns in the government's antitrust case against Microsoft—from the D.C. Circuit's stormy questioning at the two-day oral argument in late February 2001 to its affirmance of the lion's share of the government's case in June 2001, and then from the settlement between the United States and Microsoft to the continuing battle by nine states for tougher sanctions—have garnered their share of press attention.[2] But the high-profile antitrust case has not been the only Microsoft-centered controversy during the past year. Another involves the open source software movement, about which Microsoft has professed grave fears. The fears, however genuine, spring from wrongheaded myths about open source software. Exploding those myths is the burden of this Article.

In mid-February 2001, Microsoft executive James Allchin lobbed verbal grenades into the long-simmering dispute within the software development community about whether software users should have open access to a program's human-readable "source code" and the freedoms that such access can bring. The direct quotations are reproduced above. Other Microsoft executives fol-

* Assistant Professor, Lewis & Clark Law School. B.A. 1989 St. John's College; M.S. 1991, J.D. 1994 Northwestern University. I wrote this essay while the beneficiary of a summer research grant at Northwestern University Law School. Thanks to Jim Speta, Tom Merrill, Susan Davies, and John Van Voorhis (each of whom gave me valuable comments on an early draft), this essay is quite better than it would otherwise have been.

[1] Bloomberg News, *Microsoft Executive Says Linux and its Kind Threaten Innovation, at* http://investor.cnet.com/investor/news/newsitem/0-9900-1028-4825241-0.html (Feb. 14, 2001). James Allchin, group vice president of Microsoft's Platforms Group, "has overall responsibility for the product delivery, engineering and the technical architecture" of Windows and related infrastructure pieces. Microsoft, *Jim Allchin Group Vice President, at* http://www.microsoft.com/PressPass/exec/Jim/default.asp (last visited July 17, 2001).

[2] *See* United States v. Microsoft Corp., 253 F.3d 34 (D.C. Cir.), *cert. denied*, 122 S. Ct. 350 (2001). A transcript of the D.C. Circuit oral argument can be found at http://www.microsoft.com/presspass/trial/transcripts/Feb01/02-26.asp and http://www.microsoft.com/presspass/trial/transcripts/Feb01/02-27.asp.

lowed suit: Craig Mundie, Microsoft's senior vice president of Advanced Strategies,[3] denounced the open source software movement in a May 2001 speech at the NYU Stern School of Business,[4] and Microsoft CEO Steve Ballmer fumed in a June 2001 interview that "Linux is a cancer."[5] Even Chairman Gates weighed in, although he largely deferred to Mundie as "the expert" on the issue.[6]

Free and open source software devotees have not been silent. Ten leading figures in the related free and open source software movements fired back at Microsoft, jointly issuing an open rebuttal letter on May 15, 2001.[7] Furthermore, Richard Stallman, founder of the Free Software Foundation and the primary intellectual force behind the free software movement,[8] offered a more extended rebuttal in his own speech at NYU.[9] The press has dutifully covered the troop movements.[10]

"Open source is an intellectual-property destroyer," Allchin warns.[11] However, it seems most unlikely that Allchin meant that open source software literally *destroys* intellectual property. Such a

---

[3] *See* Microsoft, *Craig Mundie Senior Vice President, at* http://www.microsoft.com/PressPass/exec/craig/default.asp (last visited July 17, 2001).

[4] *See* Craig Mundie, *The Commercial Software Model, at* http://www.microsoft.com/PressPass/exec/craig/05-03sharedsource.asp (May 3, 2001).

[5] Dave Newbart, *Microsoft CEO Takes Lunch Break with the Sun-Times, at* http://www.suntimes.com/output/tech/cst-fin-micro01.html (June 1, 2001).

[6] *See* Mike Ricciuti, *Gates' Grand Design, at* http://news.com.com/2009-1082-268707.html (June 20, 2001). In an interview with CNET, Mr. Gates described the company's web services technologies and explained the basis for Microsoft's attacks on the open source software development model. *See id.*

[7] *See* Bruce Perens et al., *Free Software Leaders Stand Together, at* http://perens.com/Articles/StandTogether.html (May 15, 2001).

[8] *See* Graham Lawton, *The Great Giveaway, at* http://www.newscientist.com/hottopics/copyleft/copyleftart.jsp (last visited June 9, 2002) ("The open source movement originated in 1984 when computer scientist Richard Stallman quit his job at MIT and set up the Free Software Foundation. His aim was to create high-quality software that was freely available to everybody . . . . Stallman's move resonated round the computer science community and now there are thousands of similar projects.").

[9] *See* Transcript, Richard M. Stallman, *Free Software: Freedom and Cooperation, at* http://www.gnu.org/events/rms-nyu-2001-transcript.txt (May 29, 2001) (presenting an extended rebuttal to Microsoft's arguments and discussing the origins and development of free software).

[10] *See, e.g.*, Laurie J. Flynn, *New Economy: Despite Microsoft's Best Efforts to Kill It, the Free-Software Movement Shows No Sign of Quietly Rolling Over and Dying*, N.Y. TIMES, June 4, 2001, at C4 (noting that support for open source software continues to grow in the US and abroad, despite the efforts of Microsoft to curb the movement); *An Open and Shut Case*, ECONOMIST, May 12, 2001, at 67 (comparing proprietary and open source approaches to software development, and reviewing Microsoft's public criticisms of open source software); Andrew Leonard, *Microsoft: Free-software Licenses are the Devil's Work!*, SALON, *at* http://www.salon.com/tech/col/leon/2001/05/03/microsoft_gpl/index.html (May 3, 2001) (stating that free software does not fit into the "economic reality" of the commercial open-source market, and also suggesting that Microsoft should concentrate on its product development instead of fueling more attacks on the General Public License).

[11] Bloomberg News, *supra* note 1.

claim would, after all, be pure folly. The threat posed by open source software, he elaborated, is to "the software business and the intellectual-property business."[12] If the imbroglio between Microsoft and the free software/open source software community were merely a spat about differing business models or the relative costs of developing a software program in different ways, it might well be of little or no lasting importance to public policy. But something more fundamental appears to be in play here.

This very public disagreement centers, in fact, on consequences flowing from quite basic features of U.S. copyright law as applied to software. Additionally, by shining a light on those basic copyright law features, the disagreement provides a valuable opportunity for us to consider the appropriate level of direct public support for open source software—both in terms of how much public money (if any) should be spent obtaining open source software to power government employees' workplace computers and in terms of how often (if ever) the results of publicly-funded computer science research should be released as open source software. Allchin's professed "worry if the government encourages open source," suggests as much.[13]

The debate between proprietary, closed source software companies and free software advocates has actually been ongoing since as early as 1984, when Richard Stallman founded the GNU Pro-

---

[12] *Id.*

[13] *See* Bloomberg News, *supra* note 1. Columbia Law School Professor Eben Moglen, who has long served as the Free Software Foundation's legal counsel, makes a compelling case that the prospect of widespread government use of free software animates Microsoft's recent attacks on the free software movement. *See* Eben Moglen, *Free Software Matters: The Public's Business, at* http://emoglen.law.columbia.edu/publications/lu-09.html (Apr. 9, 2001).

Recent press reports suggest that open source software proponents have had some success in persuading governments, both in the U.S. and abroad, to fund or purchase open source software products. *See, e.g.*, Paul Festa, *Governments Push Open-Source Software, at* http://news.com.com/2100-1001-272299.html (Aug. 29, 2001); Stephen Shankland, *Korean Government Buys Into Linux, at* http://news.com.com/2110-1001-816522.html (Jan. 16, 2002); John Lettice, *MS Chief Lashes Out at German Free Software Petition at* http://www.theregister.co.uk/content/archive/23964.html (Feb. 6, 2002); John Lettice, *Report Favours Open Source, Windows Mix for Bundestag, at* http://www.theregister.co.uk/content/archive/24048.html (Feb. 13, 2002); Stephen Shankland, *Linux Contract Treads on Microsoft Turf, at* http://news.com.com/2102-1001-931027.html (June 3, 2002) (reporting that "[t]he German government has signed a deal with IBM and Linux company SuSE that makes it easier for government offices to use the open-source operating system, a move that addresses concerns about relying too heavily on Microsoft products"); Matt Loney, *EC Report Advises Open Source for Europe, at* http://zdnet.com.com/2100-1104-942055.html (July 8, 2002) (reporting the conclusions of a study sponsored by the European Commission, "called *Pooling Open-Source Software,* recommend[ing] that European administrations should share software on an open-source licensing basis, to cut soaring e-government information technology costs").

ject[14] in order to create a free software operating system. Any debate this longstanding develops myriad nuances, and the closed source/open source conflicts are no exception. With a bit of background on software development basics, however, one can readily grasp the core copyright and public funding issues at stake in this recent dispute.

Software, a computer program, is simply a set of instructions that controls a computer's operation and directs it to bring about a desired result.[15] Because software is written by people but carried out by digital computers, a given program generally exists in two forms—one called "source code" and the other called "object code."[16] Source code is the readily human-readable form of the program.[17] Object code is the machine-executable form, essentially a string of 1s and 0s.[18] Not surprisingly, a person (being a person, and not a computer) needs access to the source code if he or she wants to modify the program readily.[19] Another technological fact that has interesting copyright law consequences: when you use a piece of software on a computer, you invariably wind up making copyright-law-relevant copies in the computer's transitory memory of all or a substantial part of the program.[20] This makes software quite different from a book printed on paper: you can use the book by reading it without making a copy of it.

Imagine now that you are the author of a piece of software. You have it saved in two files, one containing the source code and the other containing the object code. It is settled law that you, as the author, hold the copyright in the software.[21] Furthermore, you control the right to copy the software (for distribution or other-

---

14 *See* Lawton, *supra* note 8.

15 The Copyright Act, for example, defines a "computer program" as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." 17 U.S.C. § 101 (2000).

16 *See* MARK LEMLEY ET AL., SOFTWARE AND INTERNET LAW 25 (2000).

17 *See id.* (describing source code as the "higher-level languages that use abbreviations and short words to convey the action [of] the program" and that "can be readily understood by skilled programmers").

18 *See id.* (describing object code as the "machine-executable form" of the software, in "binary code" of 1s and 0s).

19 Skilled computer scientists can, in fact, read code in binary form (*i.e.*, the string of 1s and 0s), but it is extremely tedious and time-consuming to do so. *See id.* at 25, 27 (noting that source code is "easier for humans to decipher"" and "it is possible to write programs directly into machine-level language"). The text thus simplifies matters a bit, but captures the points that are essential to the free software/closed source debate.

20 *See id.* at 195-96. The leading case on this point is *MAI Systems Corp.* v. *Peak Computer, Inc.*, 991 F.2d 511 (9th Cir. 1993). Congress continues to tinker, in 17 U.S.C. § 117 (2000), with the consequences of this technological fact for the relationship between software provider and software user. *See* LEMLEY ET AL., *supra* note 16, at 201-05.

21 *See* 17 U.S.C. § 201(a) ("Copyright in a work protected under this title vests initially in the author or authors of a work.").

wise), as well as the right to modify the software to produce a different program (or a "derivative work").[22] As a result, if you decide to distribute the software to others for their use, you will have choices to make about whether to allow the recipients to copy the software for redistribution or otherwise, or to modify the software into a different program. Three basic distribution approaches, each reflecting different choices about a software recipient's right to distribute copies or prepare derivative works, are of interest here.

You can put the software—source code and object code—entirely into the public domain, disclaiming all your copyright powers.[23] If you do so, anyone who receives a copy of the program thereafter can copy it as few or as many times as desired without any need for permission from you. In addition, because you have disclaimed your right to control the preparation of derivative works, the recipient can author a new piece of software comprising a modified version of your software, *e.g.*, with some additional functionalities or with some bugs fixed. Again, this can be done without any need for permission from you. Also note that a recipient or second author will hold the copyright in this resulting derivative work (at least as to the newly created material) and therefore will enjoy the same control over others' activities regarding the derivative work that you once enjoyed over your own work (before your disclaimer).[24] You will, for example, have to obtain the second author's permission before you copy or modify this new software, just like everyone else. Additionally, the second author, far from being forced to disclaim copyright protection for the new program, has the right to take a far more restrictive approach than you did to the licenses granted to others.

---

[22] *See id.* § 106 ("Subject to sections 107 through 121, the owner of copyright under this title has the exclusive right to do and to authorize any of the following: (1) to reproduce the copyrighted work in copies or phonorecords; (2) to prepare derivative works based upon the copyrighted work; [and] (3) to distribute copies or phonorecords of the copyrighted work to the public by sale or other transfer of ownership, or by rental, lease, or lending . . . .").

[23] *See* Tom W. Bell, *Escape from Copyright: Market Success vs. Statutory Failure in the Protection of Expressive Works*, 69 U. CIN. L. REV. 741, 793-94 (2001) (collecting cases and commentaries) ("Courts and commentators agree that a copyright owner can reject the copyright Act's protections and abandon an expressive work to the public domain.") (footnotes omitted) .

[24] Under the Copyright Act, derivative works are independently eligible for copyright protection. *See* 17 U.S.C. § 103(a) ("[t]he subject matter of copyright as specified by section 102 includes compilations and derivative works"), *id.* 103(b) ("[t]he copyright in a compilation or derivative work extends only to the material contributed by the author of such work, as distinguished from the preexisting material employed in the work"); Lydia Pallas Loren, *The Changing Nature of Derivative Works in the Face of New Technologies*, 4 J. SMALL & EMERGING BUS. L. 57, 61-64 (2000) (providing general discussion of derivative work right).

Alternatively, you can distribute the software in object code format only and under a license that denies the user any authority to redistribute copies of the software, to modify the software in any way, or to convert the object code into source code. This method, which one might label a "closed source" approach because it denies the user access to the source code, should look familiar. It is, in fact, the preferred distribution method among mass-market software firms such as Microsoft.

Finally, you can distribute the software with source code accompanying object code and under a license that authorizes (a) unlimited redistribution of the original package (source code, object code, and license), (b) unlimited modification of the original software, and (c) unlimited distribution of any resulting modified software provided that two simple conditions are met: First, the new software must identify both the author of the changes and the changes themselves; and second, the new software must be distributed under the same license terms and conditions as the original software. This distribution approach is known as "free software,"[25] and it comports with the definition of "open source software" promulgated in 1998.[26] The best-known free software license is the GNU Project's General Public License, or "GPL."[27] The GPL has been used to distribute numerous programs, including the GNU / Linux operating system.[28] In sharp contrast to placing a piece of

---

[25] The Free Software Foundation's web site provides the canonical definition of "free software." *See* Free Software Foundation, *The Free Software Definition*, *at* http://www.gnu.org/philosophy/free-sw.html (last visited March 5, 2002).

[26] The Open Source Initiative ("OSI") began in 1998 in an effort to make the principles of free software more palatable to corporate IT managers put off by the radical vibe that had built up around the phrase "free software": "Unlike the FSF, to which the open source philosophy is essentially a set of ethical values, OSI markets the open source concept primarily as a business model, not a model of society." Mathias Strasser, *A New Paradigm in Intellectual Property Law? The Case Against Open Sources*, 2001 STAN. TECH. L. REV. 4, 61, *at* http://stlr.stanford.edu/STLR/Articles/01_STLR_04/index.htm. For two accounts of OSI's history, see Bruce Perens, *The Open Source Definition*, and Eric S. Raymond, *The Revenge of the Hackers*, *in* OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION (Chris DiBona et al. eds., 1999).

OSI has its own canonical "Open Source Definition." *See* Open Source Initiative, *The Open Source Definition: Version 1.8*, *at* http://www.opensource.org/docs/definition.html (last visited March 5, 2002). The "Open Source Definition" has a great deal in common with the Free Software Foundation's "Free Software Definition," and OSI includes the GPL in its list of 20+ OSI-certified software licenses. *See* http://www.opensource.org/licenses/index.html (last visited March 5, 2002). The primary difference between the Free Software Definition and the Open Source Definition is that the former *requires* the author of a modified version of a program to distribute it (if at all) under the free software license, whereas the latter merely *permits* the author to do so. As a result, some open source software licenses allow authors of derivative works to take the source code of the derivative work private. *See* Perens, *supra* note 7, at 181-85 (comparing different licenses).

[27] The terms of the GPL can be found at http://www.gnu.org/copyleft/gpl.html (last visited Jan. 25, 2002).

[28] *See* http://www.opensource.org/licenses/index.html (last visited Apr. 21, 2002)

software into the public domain by utterly disclaiming copyright protection, using a free software license such as the GPL prevents downstream recipients from using the software to create new programs for distribution under a closed source approach.

The GPL demonstrates that one can harness the control that copyright law provides to make a piece of software fully and indefinitely accessible, or free, to its users. The carefully crafted license terms do all the work. In other words, open source software, far from forswearing copyright protection, relies centrally on the basic rights that copyright law gives to authors.

In the light of this insight, what sense can we make of Microsoft's statements about open source software and, more specifically, about the GPL? James Allchin, quoted above, has called open source software an "intellectual-property destroyer" and implied that it is not "American."[29] Not much to work with there. The GPL does not destroy a software author's original copyright; rather, it is predicated squarely upon it. Nor does the GPL destroy the subsequent copyright held by a user who authors a derivative software work by modifying or adding to the original program.

Allchin later clarified, through a spokesperson, that his concerns "stem from GPL paragraph 2(b)," which allegedly indicates that "anyone who adds or innovates under the GPL agrees to make the resulting code, in its entirety, available for all to use . . . [which] might constrain innovating stemming from taxpayer-funded software development.'"[30] This claim, although not absurd on its face, trades on an erroneous reading of the GPL.

To be sure, GPL paragraph 2(b) sets the price of one's right to *distribute* a derivative work at assent to a share-and-share-alike approach. According to this provision,

> You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the [original] Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.[31]

It should be noted, however, that this sharing requirement is expressly limited to works that "you distribute or publish."[32] One

---

(describing the GPL as among the "four . . . 'classic' licenses most commonly used for open-source software"); Linus Torvalds, *The Linux Edge*, *in* OPEN SOURCES, *supra* note 26, at 101, 107 (stating that Linux is licensed under the GPL).

[29] *See* Bloomberg News, *supra* note 1.

[30] Peter Coffee, *Microsoft Clarifies Exec's Open-Source Concerns*, eWEEK, Feb. 20, 2001, *available at* http://www.zdnet.com/eweek/stories/general/0,11011,2687872,00.html.

[31] Free Software Foundation, *GNU General Public License* para. 2(b), *at* http://www.gnu.org/copyleft/gpl.html (last visited Jan. 25, 2002).

[32] *Id.*

can thus innovate *for oneself, i.e.*, modify the program *for internal use,* as much as one likes and *without having to share any source code with anyone.* As the "Frequently Asked Questions about the GNU GPL" web page explains,

> The GPL does not require you to release your modified version. You are free to make modifications and use them privately, without ever releasing them. This applies to organizations (including companies), too; an organization can make a modified version and use it internally without ever releasing it outside the organization.
>
> But if you release the modified version to the public in some way, the GPL requires you to make the modified source code available to the users, under the GPL.
>
> Thus, the GPL gives permission to release the modified program in certain ways, and not in other ways; but the decision of whether to release it is up to you.[33]

In the words of the May 15 open rebuttal letter, "the legal requirements of the GPL apply only to programs which incorporate some of the GPL-covered code—not to other programs on the same system, and not to the data files that the programs operate upon."[34]

The only innovations that GPL paragraph 2(b) appears to deter, then, are those as to which a would-be innovator will not make the effort to innovate absent the ability to distribute the resulting software under the closed source approach. This is almost certainly not an empty set. It is no doubt true that some software developers will not or cannot work on improving or adapting GPL-covered software, and that the innovations these developers would have made but for the presence of the GPL might not be achieved by the developers who *are* willing to work on GPL-covered software.[35] How, then, should this fact about the GPL's apparent

---

[33] Free Software Foundation, *Frequently Asked Questions About the GNU GPL, at* http://www.gnu.org/copyleft/gpl-faq.html (last visited Jan. 25, 2002) (responding to the question "Does the GPL require that source code of modified versions be posted to the public?"); *see also* GPL para. 2 ("Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the *distribution* of derivative or collective works based on the Program.") (emphasis added); Eben Moglen, *Free Software Matters: Enforcing the GPL* 2, *at* http://emoglen.law.columbia.edu/publications/lu-12.pdf (Aug. 12, 2001) ("Almost everyone who *uses* GPL'd software from day to day needs no license, and accepts none. The GPL only obliges you if you *distribute* software made from GPL'd code, and only needs to be accepted when redistribution occurs.") (emphasis added).

[34] Perens, *supra* note 7.

[35] Of course, closing the software's source code also deters innovations, and in a similar way: Because the software's source code is closed to all developers except those chosen by the program's owner, most software developers simply cannot work on improving or adapting closed source software. Still other developers could work on the program if given the source code, but will not do so on principle. Ostensibly, Richard Stallman appears to be

incentive effects shape the government's posture toward open source software licensing models? Before exploring the question in more detail, it is worth reviewing Craig Mundie's and Steve Ballmer's aforementioned remarks.

Mundie began his speech at NYU by discussing a new Microsoft business strategy called "Shared Source,"[36] Redmond's apparent answer to the growing popularity of software covered by open source licenses, such as the GNU/Linux operating system and the Apache web server.[37] Toward the end of the speech, however, he leveled two charges at the GPL:

> The GPL mandates that any software that incorporates source code already licensed under the GPL will itself become subject to the GPL. When the resulting software product is distributed, its creator must make the entire source code base freely available to everyone, at no additional charge. [1] This viral aspect of the GPL poses a threat to the intellectual property of any organization making use of it. [2] It also fundamentally undermines the independent commercial software sector because it effectively makes it impossible to distribute software on a basis where recipients pay for the product rather than just the cost of distribution.[38]

---

such a programmer. The innovations that the excluded developers would have made but for the closed source approach might not be achieved by the developers who *are* willing and able to work on closed source software.

[36] *See* Craig Mundie, *The Commercial Software Model, at* http://www.microsoft.com/PressPass/exec/craig/05-03sharedsource.asp (May 3, 2001).

[37] *See* Stephen Shankland, *Microsoft, Red Hat Argue Open Source, at* http://news.com.com/2102-1001-270684.html (July 26, 2001) (describing Microsoft's "'shared source' plan" as "an alternative to" the GPL).

[38] Mundie, *supra* note 36 (bracketed numbers added). More recently, in an e-mail first distributed within Microsoft, Mundie echoed this second concern. In this e-mail, Mundie warned of "the potential implications of the GPL for use in disseminating the results of academic or government-funded research." Craig Mundie, *Microsoft's Views on Open Source and GPL Licensing*, at 2, *at* http://use.perl.org/article.pl?sid=01/07/26/2341206 (July 26, 2001). According to Mundie,

> [t]he GPL in this context effectively erects a wall that prevents the public and private sectors from working together. By restricting severely the rights of anyone who incorporates GPL code into their own software program, the GPL makes it impossible for commercial software companies to build on the types of academic works that have been put in the public domain and have helped fuel innovation [for] the last half-century.

Id. Similarly, at the Microsoft-sponsored April 2002 "Government Leaders' Conference," Chairman Gates opined that "GPL software is like this thing called Linux, where you can never commercialize anything around it; that is, it always has to be free." Microsoft, *Government Leaders' Conference Remarks by Bill Gates* 10-11, *at* http://www.microsoft.com/billgates/speeches/2002/04-17glc.asp (Apr. 17, 2002). But the claim that use of the GPL frustrates software commercialization or a software business sector is accurate if, and *only* if, the "commercial software companies" in question are inalterably committed to a software-as-unopenable-widget business model—rather than, for example, a software-as-opportunity-to-sell-hardware-or-maintenance-or-development-services model.

Neither criticism stands up well to scrutiny.

The first charge is groundless, no matter how one interprets the phrase "making use of it."[39] If "making use of" the software means using it to perform the task(s) for which it was designed, the claim is simply absurd. One does not, merely by running a piece of GPL-covered software, threaten the validity or enforceability of any of one's copyrights, patents, trademarks, or trade secrets. If, as is more likely, "making use of" the software means using it as the starting point for authoring a derivative software work, the claim trades on another misreading of the GPL. The only thing the GPL *requires* when you use covered software to prepare a derivative software work *without also distributing that work* is that you "cause the modified files to carry prominent notices stating that you changed the files and the date of any change."[40] As was discussed above, if you do not distribute the new software, you are not required to share it—any portion of it—with anyone. Finally, even if you do distribute the derivative work, your copyright in the work is not under any "threat."[41] Rather, it is simply conditioned on using the GPL to cover the distributed derivative work.

Mundie's second charge may or may not prove accurate, but it is certainly quite startling. According to Mundie, requiring the works derived from free software to remain free software "fundamentally undermines the independent commercial software sector."[42] Mundie essentially argues that, with a piece of free software, the author of a derivative work cannot practically price a copy of the software above the cost of reproduction and distribution. To be sure, GPL-covered software holds less promise as a platform for profitable derivative works than would the same piece of software if found in the public domain (derivatives of which can be made closed). But does this fact fundamentally undermine "the independent commercial software sector"?[43] It seems unlikely. As Andrew Leonard, a long-time student of the free software movement, observed,

> even if the most venomous interpretation of the GPL is true, there's a very simple answer for companies like Microsoft, fearful of entanglement with the stain of free software: Don't turn to open-source software at all – instead, innovate, like Microsoft's marketing literature says. Strictly written free-software licenses

---

39 Microsoft, *supra* note 36.
40 *See* Free Software Foundation, *supra* note 31.
41 *See* Mundie, *supra* note 36.
42 *Id.*
43 *Id.*

> aren't designed to make it difficult for commercial software companies to do business; they're designed to prevent corporations from profiting off of the freely donated labor of programmers without giving anything back.[44]

Microsoft's Steve Ballmer echoed his colleagues in more pungent terms. When asked whether he "view[ed] Linux and the open-source movement as a threat to Microsoft,"[45] Ballmer responded by characterizing the GPL as a cancer:

> It's good competition. It will force us to be innovative. It will force us to justify the prices and value that we deliver. And that's only healthy. The only thing we have a problem with is when the government funds open-source work. Government funding should be for work that is available to everybody. Open source is not available to commercial companies. The way the license is written, if you use any open-source software, you have to make the rest of your software open source. If the government wants to put something in the public domain, it should. Linux is not in the public domain. Linux is a cancer that attaches itself in an intellectual property sense to everything it touches. That's the way that the license works.[46]

About the only thing right here is the spelling.

"Open source is not available to commercial companies"?[47] It is, of course, *fully* available to all commercial companies—even commercial software companies. Indeed, a commercial software company can use open source software not only to carry out whatever task the software performs (and any task that the software can be modified to perform), but also to learn all that the program's source code can teach another developer about how to write good software. And merely using open source software does not force you to "have to make the rest of your software open source."[48] Instead, as should now be familiar, only a *publicly distributed* derivative work based on a GPL-covered program need itself be made open source.

When Microsoft snarls words like "destroyer," "viral" and "cancer" to describe a method of distributing software source code that relies on bedrock copyright law principles to prevent anyone from closing that code in the future, one can be forgiven for wondering

---

[44] Andrew Leonard, *Microsoft Unbound, at* http://www.salon.com/tech/col/leon/2001/06/12/monopoly_redux (June 12, 2001).

[45] *See* Newbart, *supra* note 5.

[46] *Id.*

[47] *Id.*

[48] *Id.*

(with apologies to Bertrand Russell) whether the folks in Redmond define "liberty" as the freedom to obey Microsoft.[49] However they define it, the clearest danger here for public policy is that decision-makers in government may repeat Microsoft's frequent and fundamental mistakes about the GPL and other open source software licenses.

Ostensibly, Microsoft would have us believe that open source software is in some deep and troubling way hopelessly incompatible with our intellectual property law system. This contention is false. Open source software authors, far from forswearing copyright protection, rely squarely upon it to promote and protect the freedoms they want their software users to enjoy. Indeed, the GPL and other open source software licenses enhance copyright law by demonstrating that copyright's internal logic—"the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare through the talents of authors and inventors"[50]—applies not only where personal gain takes the form of cash, but also where it takes the form of the pleasure of successfully confronting a daunting programming challenge or helping your neighbor with some software he or she needs, as well as the enhancement to your reputation in the software development community that both those activities can bring.[51]

Cash remains, of course, a perfectly good incentive for spurring software innovation. Thus it seems quite fair to ask the question that Microsoft's remarks, stripped of agitprop, suggest—namely, given that some software developers will not (or cannot) work on improving or adapting GPL-covered software, and that the innovations they would have made might not be achieved by the

---

49 Russell joked that Hegel's statist philosophy defined "liberty" as "the freedom to obey the police." Anthony Quinton, *Springtime for Hegel*, N.Y. REV. BOOKS, June 21, 2001, at 78, 79 (reviewing Terry Pinkard's *Hegel: A Biography*).

50 Mazer v. Stein, 347 U.S. 201, 219 (1954) (upholding copyright protection for statuettes that were intended for use as, and were used as, bases for mass-produced table lamps).

51 For a penetrating analysis of the point, see Stephen M. McJohn, *The Paradoxes of Free Software*, 9 GEO. MASON L. REV. 25, 35-44 (2000). For an equally penetrating, if far more skeptical, analysis, see Strasser, *New Paradigm*, *supra* note 26, at 75-87.

For my part, I am *not* arguing that only cash motivates closed source software developers, or that only altruism motivates open source software developers. Such arguments would, at the least, run aground on what we already know about service-based and other models being used to build open source software businesses. *See* Eric S. Raymond, *The Magic Cauldron*, *in* THE CATHEDRAL & THE BAZAAR 129-40 (2d ed. 2001); Sergio G. Non, *Linux Stocks Burn Out, Fade Away*, *at* http://news.cnet.com/news/0-1003-201-6947032-0.html (Aug. 24, 2001) (where a leading Linux distributor stated "Red Hat's revenue engine isn't its OS distribution; after all the Linux kernel can be had for free. The real money is in the company's Red Hat Network, used to provide patches, updates and other technical support to subscribers.").

developers who *are* willing to work on GPL-covered software, what posture should government take toward free software licensing models such as the GPL, and open source software more generally? It also seems fair, in framing an answer, to separate the two different roles that government is likely to play with respect to open source software—software purchaser, and funding source for basic computer science research and development ("R&D").

When acting as a software consumer, the Government should put open source and closed source software on an equal footing. The key questions for choosing software are (or should be) designed to identify the likely benefits and costs of making a particular choice. Does the candidate software perform well the function for which it is needed? How much will it cost to tailor it, with further development, to the system(s) already in place? How secure is the software against outside attack? How compatible is the software with other widely available hardware and software? Is the market for servicing the software competitive, and will it remain so? What future costs is one likely to incur by acquiring this software today—expensive upgrades? Root-and-branch replacement if the vendor goes out of business? What future benefits—including spillover benefits to third parties—are likely to be realized by acquiring this software today? After considering the answers to all these questions and others like them, if a government actor determines that an open source software program offers greater benefits at a given cost, then it should acquire the open source software. Likewise, if the closed source software is a better value, the government actor should acquire that software instead. In short, I can think of no good reason why government should, as a software consumer, use public money generally to favor either type of software, open or closed, as a type. Nor has Microsoft provided such a reason.

The question of public funding for computer software R&D is a bit tougher to answer. According to the traditional account of industrial R&D funding, an R&D project attracts private investment in rough proportion to the likelihood that the project's results can reliably be commercially exploited to generate a healthy return; R&D projects targeted at more basic questions, because they offer commercial benefits that are far less predictable and immediate, thus attract less private investment than applied projects.[52] This is true despite the fact that society as a whole may

---

[52] *See* John M. Golden, *Biotechnology, Technology Policy, and Patentability: Natural Products and Invention in the American System*, 50 EMORY L. REV. 101, 139 (2001) ("Investors, for their part, want a discrete and well-defined return on their investment, something that generally requires the development of a marketable product, rather than the mere disclosure of a

benefit more from having answers to more basic questions. Government thus invests public money in basic R&D to correct what would otherwise be an underinvestment in basic knowledge for widely shared benefit.[53] According to this understanding, public investments in generating basic knowledge are justified to the degree that they broaden and deepen the reserve of public information inputs on which private actors can draw for their more applied efforts. As a result, it would plainly be self-defeating for government to structure its investments of public money in basic R&D in a way that generally blocks private actors from incorporating the results of that research into their own privately funded work.[54] As Professor Lessig has observed in this context, "a government has its own interests, and closing its resources to others is not one of them."[55]

The upshot for publicly funded computer science R&D is thus difficult to determine. At first blush, the proper baseline appears to be for the government to require that the software results it funds be put into the public domain, fully disclaiming any copyright protection.[56] Once in the public domain, the advances are fully available as inputs for use by both open source and closed source software developers alike. Adherence to a public domain default rule would provide the most flexibility to downstream private actors without sacrificing any innovations that a software developer would have made if only the government-sponsored results had been provided in the distribution channel the developer pre-

---

scientific discovery. As a result, the 'funders' of industrial research and development—venture capitalists and other private investors—generally do not even seek to compete with public sector funding for basic research. They naturally focus on the development of commercial applications."). For a creative exploration of some of the shortcomings of the traditional dichotomy between basic and applied research, see Brett Frischmann, *Innovation and Institutions: Rethinking the Economics of U.S. Science and Technology Policy*, 24 VT. L. REV. 347 (2000).

53 *See generally* ROBERT COOTER & THOMAS ULEN, LAW & ECONOMICS 106-09, 126-27 (3d ed. 2000) (discussing theoretical bases for predicting that private markets undersupply public goods such as information).

54 Weapons research presents a somewhat different picture because the goal of the research—effective national defense—is a public benefit, but a large share of the research's value may come from its remaining secret (rather than from contributing to a common fund of knowledge on which others can build). My analysis thus may have little application to computer science R&D that is conducted primarily for a national security purpose.

55 LAWRENCE LESSIG, THE FUTURE OF IDEAS: THE FATE OF THE COMMONS IN A CONNECTED WORLD 247 (2001) (arguing that "the government should encourage the development of open code").

56 In effect, government-funded software R&D results would be treated the same way as works of authorship prepared by U.S. government employees as part of their official duties, which are expressly ineligible for U.S. copyright protection. *See* 17 U.S.C. § 105 (2000) ("Copyright protection under this title is not available for any work of the United States Government . . . .").

ferred (*e.g.*, open), rather than in the channel she disfavored (*e.g.*, closed).

The public domain default rule also assumes, however, that a software developer's incentives to innovate are unaffected by the existence of a concurrent development project that starts from the *same* origin (*i.e.*, the publicly funded, public domain result) but proceeds under the *opposite* distribution model. This assumption no doubt holds true in some cases. For example, Richard Stallman has described how he incorporated MIT's public-domain X Window System into his GNU free software (covered by the GPL) even though, at the same time, vendors of closed-source Unix systems were incorporating the X Window System into their own proprietary products.[57] This independence-of-development assumption, however, may not hold true in all cases. Perhaps, for a given program or family of programs, the co-existence of alternative open and closed source versions of the software would deaden the innovation-spurring incentives in both development branches. If they exist, the government in such cases would deter innovation by releasing the software into the public domain.

In an ideal world, then, the government funding source would select for each computer science R&D investment the appropriate initial licensing model. For example, the appropriate licensing model might be the public domain in instances where that model will lead to both open source and closed source innovations. On the other hand, either open or closed source would be the appropriate licensing model where the failure to pick one of the two will dead-end the project. We do not, however, live in an ideal world. I doubt that those who administer government-funded computer science R&D projects can reliably identify *ex ante* the projects that warrant a departure from the public-domain-release default rule. Moreover, my doubts stem from inherent complexities of the prediction problem (large number of variables, limited information, etc.), which are likely to remain intractable. The public domain default rule thus strikes me as the best approach for publicly funded computer science R&D in our decidedly imperfect world.

The free software/closed source debate will, of course, continue. The software development community enjoyed a live(ly) debate between Microsoft's Craig Mundie and Red Hat's Chief Technical Officer Michael Tiemann in July 2001 at the O'Reilly

---

57 *See* Richard Stallman, *The GNU Operating System and the Free Software Movement*, *in* OPEN SOURCES, *supra* note 26, at 53, 57-59; Richard Stallman, *The X Window's Trap* (1998), *at* http://www.fsf.org/philosophy/x.html (last modified Feb. 26, 2002).

Open Source Convention in San Diego, California.[58] More recently, Mundie reiterated Microsoft's objections to the GPL at the 2002 World Congress on Information Technology.[59] Whatever the forum, the debate should focus on the facts rather than on myths or groundless fears.

---

[58] *See* Shankland, *supra* note 37.

[59] *See* Bruce Perens, *Deciphering the Open-Source War*, *at* http://news.com.com/2010-1078-855155.html (Mar. 8, 2002).