FOR BETTER OR WORSE: INTRODUCING THE GNU GENERAL PUBLIC LICENSE VERSION 3

By John Tsai

I. INTRODUCTION

"I designed the GNU General Public License for a very simple purpose: to defend the freedom of every user of a free program."[1] Thus, Richard Stallman, creator of the free software movement and founder of the non-profit Free Software Foundation ("FSF"), defines the vision for the GNU General Public License ("GPL").[2] Version 2 of the GPL ("GPLv2") was released in 1991[3] and is now one of the most popular free and open source software ("FOSS") licenses in the world.[4]

On June 29, 2007, the FSF released version 3 of the GPL ("GPLv3").[5] GPLv3 attempted to address both the legal and technological changes that occurred after GPLv2's release in 1991. During this period, lawyers and scholars raised legal uncertainties about GPLv2, questioning the license's enforceability and scope. This Note analyzes the perceived shortcomings of GPLv2, discusses how GPLv3 addresses those issues, including patent

1. Richard Stallman, RMS Announces Release of GPLv3, http://gplv3.fsf.org/rms_gplv3_launch_transcript (last visited Apr. 4, 2008).

2. Free software is a software movement started by Stallman in the 1970's. For a brief history, *see infra* Section II.A.

3. Robert W. Gomulkiewicz, *A First Look at General Public License 3.0*, COMPUTER & INTERNET LAWYER, Nov. 2007, at 15.

4. Ronald J. Mann, *Commercializing Open Source Software: Do Property Rights Still Matter?*, 20 HARV. J. LAW & TECH. 1, 11 (2006). While the words free software and open source are often interchangeably used (and the respective movements share many common goals), the terms differ in one significant way: free software refers to political movement, while open source refers to a software development methodology. *See infra*, Sections II.B, II.C; *see also* Richard Stallman, Why Open Source Misses the Point of Free Software, http://www.gnu.org/philosophy/open-source-misses-the-point.html (last visited Apr. 4, 2008). Therefore, when this Note refers to the aims and characteristics of both movements, it uses the term "FOSS," an acronym for free and open source software. To signify the characteristics and aims of one group and not the other, this Note will use the individual terms where appropriate.

5. Stallman, *supra* note 1. The full text of GPLv3 is available at GNU General Public License, http://www.gnu.org/licenses/gpl-3.0.html (last visited Apr. 4, 2008) [hereinafter GPLv3]. The full text of GPLv2 is available at GNU General Public License v2.0, http://www.gnu.org/licenses/old-licenses/gpl-2.0.html (last visited Apr. 4, 2008) [hereinafter GPLv2].

licenses, and evaluates the changes' impact on both Stallman's "social" free software movement and the more "practical" open source movement.

This Note argues that while GPLv3 has made progress in clarifying GPLv2's legal uncertainties, the new version still contains ambiguities in the scope of the license grant. These ambiguities, along with GPLv3's hostility toward software patents, should make the technology industry and open source communities cautious about adopting GPLv3. Further, several ideologically motivated provisions in GPLv3 will exacerbate the differences between the free software and the open source movements. Businesses looking for GPLv3 to clearly delineate their legal obligations will find little comfort.

Part II of this Note examines the history of the GPL and the development of the free software movement. Part III discusses perceived problems of GPLv2—both ambiguities in the scope of the copyright license grant and the lack of express provisions governing patents. Part IV examines the implications of new language in GPLv3 with respect to copyrights and patents. Finally, Part V discusses initial reactions to GPLv3 within various software communities, identifies obstacles to relicensing under GPLv3, and points out unsettled issues regarding the new version of the license.

II. HISTORICAL BACKGROUND OF THE GPL

Before examining GPLv3 in detail, it is helpful to touch on the history of the GPL and accompanying software movements. Section II.A explains how Richard Stallman's personal project gave birth to the free software movement. Section II.B summarizes the creation of the GPL. Section II.C recounts the development of the more pragmatic open-source software movement. Finally, Section II.D traces adoption of the GPL, which is now used by thousands of open source software projects worldwide.[6]

6. *See* STEVEN WEBER, THE SUCCESS OF OPEN SOURCE 66 (2004) (citing statistics from SourceForge, Counter.li.org, and the Orbiten Free Software Survey); Sourceforge.Net: Software Map, http://sourceforge.net/softwaremap/trove_list.php?form_cat=18 (last visited Apr. 4, 2008) (showing over 120,000 registered software projects on one of the largest online repositories of open source software, SourceForge.Net, as of April 2008). FOSS software is typically developed within projects, in which a group of software developers is organized around a core source code base. *See id.* at 62-65 (explaining the model and noting variations with regard to formality, hierarchy, and stability).

A. Richard M. Stallman and the Birth of Free Software

The free software movement began in the 1970's when Richard M. Stallman worked as a programmer at MIT's Artificial Intelligence lab.[7] Stallman decided to solve a problem with the lab's centralized printer: paper jams.[8] With access to the printer's software source code,[9] Stallman modified the printer software so that it would notify all lab members when the printer jammed.[10] When the lab received a new Xerox printer, Stallman tried to improve it in the same manner.[11] However, Xerox would not release the printer's source code.[12] Stallman's encounter with this proprietary software model marked the beginning of his vision of the free software movement.[13] He believed proprietary software was fundamentally incompatible with his conception of the "golden rule."[14] For Stallman, sharing source code was, and is, a moral obligation.

In response, Stallman decided to create a computing environment where he could guarantee that the source code would always be available, or "free."[15] When coining the phrase "free software," Stallman meant "free" as in freedom to study and modify source code, not "free" in price.[16] FSF's catchphrase for this concept is "free speech, not free

7. SAM WILLIAMS, FREE AS IN FREEDOM: RICHARD STALLMAN'S CRUSADE FOR FREE SOFTWARE 1-12 (2002), *available at* http://www.oreilly.com/openbook/freedom; Brian W. Carver, Note, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443, 444 (2005).

8. WILLIAMS, *supra* note 7, at 1-5.

9. When a program, such as Microsoft Word, runs on a computer, the program file is an executable file in machine-readable binary format, called object code. The file contains a set of instructions understandable to a computer, but not software programmers. By contrast, source code represents a human-readable version of a program. Software developers write source code, not object code. A program called a compiler is used to translate source code into object code. *See* JOHN L. HENNESSY & DAVID A. PATTERSON, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE/SOFTWARE INTERFACE 5-9 (2d ed. 1998).

10. WILLIAMS, *supra* note 7, at 1-5.

11. *Id.* at 4-9.

12. *Id.*

13. *Id.* at 10. Proprietary software refers to "software for which no access to the source code is provided." Carver, *supra* note 7, at 445 n.13 (citing Jonathan Zittrain, *Normative Principles for Evaluating Free and Proprietary Software*, 71 U. CHI. L. REV. 265, 271 (2004)).

14. *See* Carver, *supra* note 7, at 445 n.17 (citing Initial Announcement, http://www.gnu.org/gnu/initial-announcement.html (last visited Apr. 4, 2008)).

15. *Id.*

16. GPLv2, *supra* note 5, at pmbl.; GPLv3, *supra* note 5, at pmbl.

beer."[17] At that time, the standard computer operating system was UNIX, developed by AT&T Bell Laboratories.[18] Stallman's own project was to write an entire operating system environment compatible with UNIX, called "GNU," a recursive acronym for "GNU's Not UNIX."[19] In 1985, Stallman also founded the FSF, a "tax-exempt charity for free software development."[20] The FSF's goals are primarily social and political, not technical or economic, which is key to understanding the distinction between Stallman's free software movement and the open source movement described below.[21]

B. Creation of the GNU GPL

Although Stallman was initially weary of copyright licenses, he eventually recognized that he could use licenses to preserve the freedoms of his software by ensuring that others who modified or distributed his source code would be bound by the licenses' terms.[22] Stallman began to experiment with different copyright licenses to ensure sharing of his source code.[23] These eventually evolved into the first version of the GPL, which

17. RICHARD M. STALLMAN, *The Free Software Definition*, in FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN 43 (Joshua Gay ed., 2002) (introducing the Free Software Definition, a list of four criteria for a "free software" license), *available at* http://www.gnu.org/philosophy/fsfs/rms-essays.pdf.

18. WILLIAMS, *supra* note 7, at 90-91.

19. *Id.* at 89-90.

20. *Id.* at 106; Douglas A. Hass, Note, *A Gentlemen's Agreement: Assessing the GNU General Public License and Its Adaptation to Linux*, 6 CHI.-KENT J. INTELL. PROP. 213, 215 (2007) (discussing the creation and purpose of the FSF).

21. Eben Moglen & Richard Stallman, GPL Version 3: Background to Adoption, http://www.fsf.org/news/gpl3.html (last visited Apr. 4, 2008) (stating the FSF's "goals are primarily social and political, not technical or economic").

22. WILLIAMS, *supra* note 7, at 123-24. Copyright in code, combined with licensing of code, ensures the freedom of software. Software is protected by copyright law. The moment a programmer writes code, copyright law gives the programmer a bundle of exclusive rights, including the rights to copy, distribute, and create derivative works. The programmer can give away these exclusive rights via licensing. Free software and open source programmers can therefore use copyright to license their code with specific conditions, such as making the source code of any modifications available. Robert Gomulkiewicz, *General Public License 3.0: Hacking the Free Software Movement's Constitution*, 42 HOUS. L. REV. 1015, 1022-23 (2005). For more information on open source copyright licensing, see generally LAWRENCE ROSEN, OPEN SOURCE LICENSING: SOFTWARE FREEDOM AND INTELLECTUAL PROPERTY LAW (2004), *available at* http://www.rosenlaw.com/oslbook.htm; ANDREW M. ST. LAURENT, UNDERSTANDING OPEN SOURCE AND FREE SOFTWARE LICENSING (2004), *available at* http://www.oreilly.com/catalog/osfreesoft/book.

23. WILLIAMS, *supra* note 7, at 124.

Stallman published in 1989.[24] Two years later, Stallman published GPLv2.[25] The GPL was (and is) a "copyleft," or reciprocal license: any derivative work of a copyleft-licensed work must be licensed under the same license.[26] Downstream licensees, no matter how far removed from the original licensor, are thus bound by the key GPL terms.[27]

C. The Open Source Movement

Although Stallman started a social movement based on his philosophy of free software, others focused on the practical benefits in providing source code to users.[28] The open source movement's founders worried that businesses were not embracing free software because they confused the word "free" with no cost, despite Stallman's efforts to associate "free" with freedom.[29] The movement coined the term "open source" to encourage companies to adopt an open development process, which movement leaders (such as Eric Raymond)[30] saw as having distinctly practical bene-

24. *Id.* at 126. Stallman's vision of free software rested on four fundamental freedoms: (1) the freedom to run the program, for any purpose; (2) the freedom to modify the program to suit one's needs (meaning access to the source code); (3) the freedom to redistribute copies, either gratis or for a fee; and (4) freedom to distribute modified versions of the program, so that the community can benefit from one's improvements. These freedoms are codified in the Free Software Definition, *see supra* note 17.

25. GPLv2, *supra* note 5.

26. RICHARD M. STALLMAN, *What is Copyleft?*, *in* FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN, *supra* note 17, at 91-92; *see also* Carver, *supra* note 7, at 453, 455-56; Hass, *supra* note 20, at 217. Copyleft licenses have been described as being "viral" because any software which incorporates GPL code must itself be licensed under GPL under copyleft terms. *See, e.g.*, WILLIAMS, *supra* note 7, at 14-16; Craig Mundie, Chief Research and Strategy Officer of Microsoft, Address Before the New York University Stern School of Business, May 3, 2001, http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.mspx (last visited Apr. 4, 2008) (stating that the "viral aspect of the GPL poses a threat to the intellectual property of any organization making use of it").

27. Carver, *supra* note 7, at 455-56; *see also* Christian H. Nadan, *Open Source Licensing: Virus or Virtue?*, 10 TEX. INTELL. PROP. L.J. 349, 357-58 (2002) (explaining that the copyleft nature of the GPL ensures that downstream recipients of GPL-licensed code remain bound to the terms of the license).

28. For more information on the open source movement, *see* ERIC RAYMOND, THE CATHEDRAL AND THE BAZAAR: MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY (2d ed. 2001).

29. Carver, *supra* note 7, at 449-50; Nadan, *supra* note 27, at 354-55.

30. *See generally* WILLIAMS, *supra* note 7, at 159-169 (tracing the origins and trajectory of the open source movement); RAYMOND, *supra* note 28, at 169-191 (discussing Raymond's role as the "accidental revolutionary" of the open source movement and the open source philosophy generally).

fits, such as the ability to find and fix bugs more quickly.[31] Rather than following the free software movement, which argued that software should never be proprietary, the open source movement sought to cooperate with the business world.[32] The open source movement has resulted in the creation of numerous open source licenses, such as the Apache License (under which the ubiquitous Apache web server is licensed) and the Mozilla Public License (under which the Mozilla Firefox web browser is licensed).[33]

D. Adoption of the GPL

The GPL has been widely adopted. As of April 2008, almost 30,000 of roughly 44,000 software projects on Freshmeat.net, and over 80,000 of about 117,000 open source software projects on Sourceforge.net, two of the most popular online repositories of open source software, are licensed under the GPL.[34] Since December 1991, Linux has been licensed under the GPL, and Linux distributions also incorporate many GNU tools.[35]

The GPL has flourished as an open source license for three reasons. First, the philosophy that Stallman outlined appealed to many software developers and users just as the Internet emerged and allowed users to collaborate over large geographical distances because it "facilitate[d] innova-

31. Carver, *supra* note 7, at 449-50; Nadan, *supra* note 27, at 354-55. Note that the term "open source" and "free software," while having many similarities, are not the same. *See* Carver, *supra* note 7, at 450-53 (noting the differences between the Free Software Definition and the Open Source Definition). Richard Stallman is opposed to using the term "open source" for ideological reasons. Stallman, *supra* note 4, para. 6-9.

32. Nadan, *supra* note 27, at 354 ("'We realized it was time to dump the confrontational attitude that has been associated with 'free software' in the past and sell the idea strictly on . . . pragmatic, business-case grounds[.]'" (quoting Open Source Initiative, History of the OSI, http://www.opensource.org/history (last visited Apr. 4, 2008)) (alterations in original).

33. Apache License, Version 2, http://www.apache.org/licenses/LICENSE-2.0 (last visited Apr. 4, 2008); How the ASF Works, http://www.apache.org/foundation/how-it-works.html (last visited Apr. 4, 2008); Mozilla Public License Version 1.1, http://www.mozilla.org/MPL/MPL-1.1.html (last visited Apr. 4, 2008); About Mozilla, http://www.mozilla.org/about/ (last visited Apr. 4, 2008).

34. *See* freshmeat.net: Statistics and Top 20, http://freshmeat.net/stats (last visited Apr. 4, 2008); SourceForge.net: Software Map, http://sourceforge.net/softwaremap/trove_list.php?form_cat=14 (last visited Apr. 4, 2008) (listing all OSI-approved licensed projects at SourceForge); SourceForge.net: Software Map, http://sourceforge.net/softwaremap/trove_list.php?form_cat=15 (last visited Apr. 4, 2008) (listing GPL projects only).

35. WILLIAMS, *supra* note 7, at 137-38 (outlining the development of GNU/Linux and the open source movement).

tion and improvements with and within the software."[36] Second, software licensed under the GPL was usually provided at no cost or very low cost to cover the expense of a copy.[37] Lastly, the GPL's reciprocal requirement that any distributed modifications must themselves be licensed under the GPL helped to perpetuate both the license and the licensed software.[38]

III. LEGAL ISSUES WITH GPL VERSION 2

While GPLv2 has become a popular FOSS license, it is far from perfect. Academics, members of the free software and open source communities, and Stallman himself perceived many problems and shortcomings with GPLv2 in the sixteen years since its release in 1991.[39] One problem that puzzled lawyers was the ambiguous scope of GPLv2's license grant.[40] Another concern was whether the GPLv2 granted an implied patent license.[41] This section discusses the pressing legal risks that lawyers and businesses have faced when confronted with GPLv2.

36. Richard Stallman, *The GNU Operating System and the Free Software Movement*, in OPEN SOURCES: VOICES FROM THE OPEN SOURCE REVOLUTION 53, 56, 60 (Chris DiBona et al. eds., 1999).

37. Stallman used to charge $150 for a copy of his EMACS software to support himself, but encouraged those who received copies to share them with others. Stallman, *supra* note 36, at 58.

38. *Id.* at 62, 66.

39. *See, e.g.*, ROSEN, *supra* note 22, at 109-140 (analyzing GPLv2 in detail and raising potential legal problems with ambiguities in the license terms); WEBER, *supra* note 6, at 213 (pointing out problems with GPLv2's warranty provisions as well as the license's binding effect on third parties); Lothar Determann, *Dangerous Liaisons—Software Combinations as Derivative Works?*, 21 BERKELEY TECH. L. J. 1421, 1480-96 (2006) (noting the problems with GPLv2 and recommending changes for GPLv3); Gomulkiewicz, *supra* note 22, at 1028-36 (raising several problems with GPLv2); Robert W. Gomulkiewicz, *De-bugging Open Source Software Licensing*, 64 U. PITT. L. REV. 75, 83-92 (2002) (discussing several flaws and recommending a process to fix them); Mitchell L. Stoltz, Note, *The Penguin Paradox: How the Scope of Derivative Works in Copyright Affects the Effectiveness of the GNU GPL*, 85 B.U. L. REV. 1439 (2005) (describing the problems in the scope of derivative works); Richard Stallman, Why Upgrade to GPL Version 3, http://gplv3.fsf.org/rms-why.html (last visited Apr. 4, 2008) (discussing the various "existing problems" that GPLv3 will fix).

40. The scope of the license grant hinged on the scope of derivative works in copyright for software. This problem is discussed below in detail, *see infra* Section III.A.

41. *See* Gomulkiewicz, *supra* note 22, at 1033-34 (querying whether GPLv2 contains an implied patent license); Stallman, *supra* note 39 ("With GPLv2, users rely on an implicit patent license. . . .").

A. Scope of GPLv2's License Grant and Copyleft Obligation

Copyright gives the author of a piece of software the exclusive right to copy, distribute, and prepare derivative works.[42] GPLv2 grants users the right to copy, distribute, and modify GPLv2 programs conditioned upon two important obligations.[43] First, the user must distribute that work and any derived works under the terms of GPLv2, per section 2, paragraph b:

> 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 the License.[44]

Second, the user must include the program's source code (or include a written offer to supply such source code) with any redistribution of the program in object code format under section 3 of GPLv2.[45] These provisions are the core of GPLv2's "copyleft" obligation: to license the original and modified versions of the program under the same license terms, while providing the source code to all downstream licensees.

The key ambiguity in GPLv2 is how far this copyleft obligation should reach. The issue originates from the use of the term "derivative work" in GPLv2. Section 0 of the license defines a "work based on the program," as "either the Program, or any derivative work under copyright law: *that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language*."[46] This provision has several possible legal interpretations.

---

42. 17 U.S.C. § 106 (2000) (listing the exclusive rights granted under copyright). Software is regarded as a "literary work" protected under the U.S. Copyright Act. 17 U.S.C. § 101 (2000) (defining "literary works" broadly); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249 (3d Cir. 1983) ("[A] computer program, whether in object code or source code, is a 'literary work' and is protected from unauthorized copying").

43. GPLv2, *supra* note 5, §§ 1-3.

44. *Id.* § 2(b).

45. *Id.* § 3.

46. *Id.* § 0 (emphasis added). GPLv2 section 0 defines key terms and outlines the scope of the license:

> This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program," below, refers to any such program or work, and a "work based on the Program" means either the Program, or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language.

Id.

One interpretation is that the scope of derivative works subject to the copyleft obligation mirrors the scope of derivative works under U.S. copyright law. U.S. copyright law defines a derivative work as:

> [A] work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a "derivative work."[47]

The definition above does not specifically address software, and some commentators have pointed out the unclear scope of a "derivative work" in software under statute and case law.[48]

However, certain terms in GPLv2 indicate that the scope of works to which the copyleft obligation applies is broader than the scope of derivative works under U.S. copyright law. Section 0 of GPLv2 adds an interpretive gloss to the meaning of "derivative work" by including within the definition "a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language."[49] When one copyrighted work is contained in a portion of another, it is considered a collective work or compilation. The U.S. Copyright Act defines these terms as follows:

> A "collective work" is a work, such as a periodical issue, anthology, or encyclopedia, in which a number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole.
>
> A "compilation" is a work formed by the collection and assembling of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship. The term "compilation" includes collective works.[50]

47. 17 U.S.C. § 101 (2000).

48. *See* Determann, *supra* note 39, at 1427-28 (noting lack of court-developed rules for classifying software combinations as derivative works); Carver, *supra* note 7, at 458 (noting that neither statutes nor case law make clear what constitutes a derivative work for software); Stoltz, *supra* note 39 (discussing how the current state of software derivative works affects GPLv2).

49. *See* text accompanying note 46.

50. 17 U.S.C. § 101 (2000). One way to think about the distinction between collective works and compilations is that collective works are creative collections of copy-

Based on these definitions, the italicized proviso appears to broaden the scope of GPLv2 to include compilations and collective works under U.S. copyright law. For example, if someone "assembled" preexisting GPLv2-licensed source code into a larger program, the GPLv2 code might be considered "contained" in the larger program (which is itself a copyrighted work). In the explanatory notes under Section 2, GPLv2 also explicitly includes "collective works" as triggering the copyleft obligation.[51] The license language appears to merge derivative works and collective works together, which is troublesome given that these terms have very distinct definitions under U.S. copyright law.[52] The puzzling result is a potential headache for proprietary software companies.[53] The viral nature of GPLv2 comes from its requirement that software "contain[ing] . . . any part of" GPLv2-licensed code must also be distributed under the license.[54] Incorporating GPLv2-licensed source code into the source code of a proprietary software product could potentially "infect" the proprietary software, causing the proprietary source code to be automatically licensed under GPLv2.[55] Because the proprietary software now "contains" the

righted works: both the collection and the collected works must be creative in nature. Compilations include both creative collections of creative works (collective works) and creative compilations of non-creative materials (such as names and phone numbers). *See* Determann, *supra* note 39, at 1431 (citing Feist v. Rural Tel. Serv. Co., 499 U.S. 340, 362-64 (1991)).

51. GPLv2, *supra* note 5, § 2, para. 5.

52. *See* Determann, *supra* note 39, at 1431. Determann succinctly distinguishes compilations, collective works, and derivative works under U.S. Copyright law by stating:

> In the case of a compilation, the existing material remains intact and unchanged, and the "combination creativity" remains separate and clearly distinguishable from the existing material. In the case of a non-derivative new work, existing material may be remotely reflected in the new work, but its contribution is insubstantial. The derivative work category lies somewhere in the middle: existing creative material constitutes a substantial part of the new derivative work, and the new creative material appears in the form of inseparable changes to the existing material.

Id.

53. *See* Nadan, *supra* note 27, at 359-60 (outlining the potential consequences of a scenario where one of thousands of Microsoft developers incorporate a few lines of GPLv2 code into Microsoft Windows).

54. GPLv2, *supra* note 5, § 2(b).

55. Nadan, *supra* note 27, at 359 ("Thus, if you incorporate some GPL code in your proprietary software product, arguably your whole proprietary product becomes open source and must be licensed by you under the GPL."); Greg R. Vetter, *"Infectious" Open Source Software: Spreading Incentives or Promoting Resistance?*, 36 RUTGERS L.J. 53,

GPLv2-licensed code, the entire program might potentially "be licensed as a whole at no charge to all third parties under the terms" of GPLv2.[56] To complicate matters, GPLv2 uses inconsistent terminology, sometimes referring to "modified" work or "modification" instead of "derivative work" or "work based on the Program."[57]

1. Dynamic Linking in Software

The technical nature of software makes the ambiguities in GPLv2 particularly problematic. One technical issue in GPLv2 involves the software principle of "linking." Software programs are often modularized into smaller, more manageable subcomponents because programs tend to be large and complex.[58] Each smaller subcomponent can be more easily written and understood by individual programmers, and work on different subcomponents can be accomplished in parallel.[59] Modularization also allows programmers to easily reuse code,[60] a key part of FOSS development.[61] Software "libraries" are subcomponents of general-purpose, reusable code that perform common functions, such as printing, reading data from a disk, or opening a file.[62] These modularized subcomponents, which are often separate programs or files, can then be combined together, or "linked," to create a larger program.[63]

Linking can be accomplished two ways: statically or dynamically. Static linking involves embedding the object code of a subcomponent directly into the object code of the larger program, like copying chapters from one book into another.[64] In contrast, a dynamically linked subcomponent exists as an independent set of object code that is simply referenced by the code of the larger program, much like web page, or a cross reference from one book to another.[65] Dynamic linking also allows for

88-94 (2004) (discussing the problems confronted by programmers in attempting to interpret section 2(b)).

56. *See* GPLv2, *supra* note 5, § 2(b).

57. *Id.* §§ 2, 5.

58. *See* W. P. Stevens, G. J. Myers, & L. L. Constantine, *Structured Design*, 38 IBM SYSTEMS J. 231, 232 (1999), *available at* http://www.research.ibm.com/journal/sj/382/stevens.pdf.

59. *See id.* at 254-55.

60. *See id.*

61. WEBER, *supra* note 6, at 75-76 (describing how open source facilitates reuse of code to minimize "reinventing the wheel").

62. HENNESSY & PATTERSON, *supra* note 9, at 8.

63. *Id.* at 158-59, A-17.

64. ABRAHAM SILBERSCHATZ ET AL., OPERATING SYSTEM CONCEPTS 278 (6th ed. 2002).

65. *Id.*

more efficient use of disk space and memory.[66] Because a dynamically linked subcomponent runs independently from the larger program, other programs can share the same dynamically-linked subcomponent simultaneously.[67]

Static linking almost certainly falls within the scope of GPLv2's copyleft obligation. It embeds a subcomponent's object code into the larger program, creating a derivative work under GPLv2 and thereby "infecting" the larger program.[68] Because the process of dynamic linking is different, some have argued that a proprietary program could dynamically link to a GPLv2-licensed subcomponent without creating a derivative work subject to GPLv2's copyleft obligation.[69] However, the FSF insists that dynamic linking to a GPLv2-licensed library can create a single combined work licensed under GPLv2.[70] The FSF fashioned a new license, called the GNU Lesser General Public License ("LPGL") to allow for GPL subcomponents to be dynamically linked with proprietary pro-

66. *Id.*

67. IBM, AIX 6.1 PERFORMANCE MANAGEMENT 338 (2007), http://publib.boulder.ibm.com/infocenter/systems/topic/com.ibm.aix.prftungd/doc/prftungd/prftungd.pdf (last visited Apr. 4, 2008).

68. This is the "viral" effect of GPLv2. *See supra* Section III.A.

69. *See* Determann, *supra* note 39, at 1458-62, 1488 n.255 (stating "dynamic linking" of libraries "would not qualify as derivative works"); Stoltz, *supra* note 39, at 1456-59, 1463-64 (discussing the "linking exemption" to the derivative work right and how it applies to GPLv2).

70. *See, e.g.*, The LPGL and Java, http://www.gnu.org/licenses/lgpl-java.html (last visited Apr. 4, 2008); GNU GPLv2 Frequently Asked Questions, Mere Aggregation, http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html#MereAggregation (last visited Apr. 4, 2008). The FSF, in describing what is considered a software combination under the terms of GPLv2, states:

> What constitutes combining two parts into one program? This is a legal question, which ultimately judges will decide. We believe that a proper criterion depends both on the mechanism of communication (exec, pipes, rpc, function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are interchanged). If the modules are included in the same executable file, they are definitely combined in one program. If modules are designed to run linked together in a shared address space, that almost surely means combining them into one program. By contrast, pipes, sockets and command-line arguments are communication mechanisms normally used between two separate programs. So when they are used for communication, the modules normally are separate programs. But *if the semantics of the communication are intimate enough, exchanging complex internal data structures*, that too could be a basis to consider the two parts as combined into a larger program.

Id. (emphasis added).

grams.[71] Complicating the analysis, GPLv2 provides a "special exception" to the copyleft obligation for certain system libraries that link to the major components of an operating system.[72] While this was meant to exempt distribution of essential proprietary libraries used in compiling GPL-based applications, it is unclear how broadly the exception can be interpreted.[73]

2. *Linux Kernel Modules*

Another technical issue in GPLv2 concerns Linux kernel modules.[74] The kernel is the basic program at the heart of an operating system.[75] Linux offers an easy process to add functionality to the kernel: modules.[76] Linux kernel modules are subcomponents that can be individually compiled and dynamically linked, or "loaded," into the kernel through an interface,[77] avoiding the need to recompile the entire kernel and reboot the system when installing software such as printer drivers.[78]

---

71. GNU Lesser General Public License, Version 2.1, http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html (last visited Apr. 4, 2008). The newest version of the LGPL is version 3, which was released simultaneously with GPLv3. GNU Lesser General Public License, Version 3, http://www.gnu.org/licenses/lgpl.html (last visited Apr. 4, 2008). Ultimately, the FSF does not recommend the use of LPGL, as Stallman believes it breeds a dependence on proprietary software. *See* Richard Stallman, Why You Shouldn't Use the Lesser GPL For Your Next Library, http://www.gnu.org/licenses/why-not-lgpl.html (last visited Apr. 4, 2008).

72. *See* GPLv2, *supra* note 5, § 3, para. 2. GPLv2 section 5 explains the exception to the copyleft obligation:

> For an executable work, complete source code means all the source code for all the modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a *special exception*, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless the component itself accompanies the executable.

Id. (emphasis added).

73. *See id.* A compiler is a program that translates source code into object code. *See supra* note 9 and accompanying text.

74. For more technical information on Linux kernel modules, *see* ALESSANDRO RUBINI ET AL., LINUX DEVICE DRIVERS (3d ed. 2005), *available at* http://lwn.net/Kernel/LDD3.

75. The kernel is the core program that manages all system resources and interacts directly with computer hardware. SILBERSCHATZ ET AL., *supra* note 64, at 6, 696.

76. RUBINI ET AL., *supra* note 74, at 5.

77. Programs are often made up of smaller subcomponents. *See supra* Section III.A.1.

78. RUBINI ET AL., *supra* note 74, at 5.

Since the Linux kernel is licensed under GPLv2, the critical concern is whether a dynamically linked kernel module constitutes a derivative work under the license. The FOSS community is deeply divided on this issue. Linus Torvalds, creator of the Linux kernel, has distinguished loadable kernel modules from dynamic linking, stating that while he does not like closed source kernel modules, he accepts them.[79] Conversely, Eben Moglen, former general counsel of the FSF and a Columbia Law School professor, has stated that "that non-GPL, non-free loadable kernel modules represent GPL violations."[80] He believes closed source kernel modules are simply exceptions allowed by Torvalds and other Linux developers.[81] Reconciling these positions is a difficult endeavor.

Some companies have exploited this kernel module uncertainty by releasing closed source, or proprietary drivers for their hardware. For example, two graphic card chipmakers, Nvidia and ATI Technologies, release their driver modules solely in closed source format.[82] ATI does so for intellectual property reasons, as their drivers contain third-party intellectual property as well as trade secrets.[83] While intellectual property is a consideration for Nvidia, it says that a graphics driver is so complicated that open sourcing it "would not help."[84] In any case, Linux distributors Red Hat and Novell are beginning to shun proprietary drivers.[85] If the Linux kernel community demands the end of closed source Linux drivers, companies may be discouraged from releasing proprietary Linux drivers for intellectual property or other reasons, which would ultimately hurt Linux's popularity.

79. For a summary of Torvalds' views on kernel modules, see Proprietary Kernel Modules, http://linuxmafia.com/faq/Kernel/proprietary-kernel-modules.html (last visited Apr. 4, 2008).

80. Joe Brockmeier, *Interview: Eben Moglen*, LINUX WEEKLY NEWS, Aug. 10, 2005, http://lwn.net/Articles/147070 (last visited Apr. 4, 2008).

81. *Id.*

82. Steven Shankland, *New Linux Fuels Old Debate*, CNET NEWS.COM, April 17, 2006, http://www.news.com/New-Linux-look-fuels-old-debate/2100-7344_3-6061491.html (last visited Apr. 4, 2008). Nvidia packages their closed source driver with an open source interface to the kernel code. *Id.*

83. *Id.* Software source and object code can be protected as trade secrets. Trandes Corp. v. Guy F. Atkinson Co., 996 F.2d 655, 663-64 (4th Cir. 1993).

84. Shankland, *supra* note 82.

85. *Id.*

B. Software Patents Under GPLv2

Software patents were very rare in 1991 when GPLv2 was released.[86] Now, more than fifteen years later, software parents have become common in the U.S. software industry.[87] FOSS software is not immune from patent infringement suits.[88] FOSS advocates have proposed a variety of methods to combat the potential threat of software patents on open source software.[89] Besides the fundamental ideological schism between the FOSS community's collaborative approach and software patent monopolies, the threat of patent infringement has been an ever-present concern for the community.[90]

GPLv2's preamble voices Stallman's opposition to patents, stating that "any free program is threatened constantly by software patents."[91] However, the GPLv2 lacked an express patent license. GPLv2's section 7 requires anyone distributing a GPLv2-licensed program to allow "royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you."[92] Thus, if a distributor obtained a patent license that prevented royalty-free distribution to all downstream recipients, then he must not distribute the code at all.[93] Likewise, a patent holder who distributes GPLv2 software must permit royalty-free redistribution to all

86. The U.S. Patent and Trademark Office (USPTO) did not start issuing standards for software patents until 1996. *See* ROBERT PATRICK MERGES & JOHN FITZGERALD DUFFY, PATENT LAW AND POLICY: CASES AND MATERIALS 154-55 (4th ed. 2006) (citing U.S. PTO, *Examination Guidelines for Computer-Related Inventions*, 61 Fed. Reg. 7478 (1996)). However, there remained uncertainty regarding the validity of software patents until the Federal Circuit's State Street Bank decision. *See* State Street Bank v. Signature Financial Group, Inc., 149 F.3d 1368 (Fed. Cir. 1998) (allowing machine and process claims covering software).

87. Robert P. Merges, *Software and Patent Scope: A Report From the Middle Innings*, 85 TEX. L. REV. 1627, 1641 (2007). One recent paper found that more than 20,000 software patents are granted each year. James Bessen & Robert M. Hunt, *An Empirical Look at Software Patents*, 16 J. ECON. & MGMT. STRATEGY 157, 158 (2007).

88. In fact, the first patent infringement suit against open source software was filed in October 2007, by a patent holding company called IP Innovation and Technology against Red Hat and Novell. Paul McDougall, *Red Hat, Novell Sued Over Linux Patents*, INFOWEEK.COM, Oct. 12, 2007, http://www.informationweek.com/news/showArticle.jhtml?articleID=202402004 (last visited Apr. 4, 2008). Microsoft claims that Linux infringes on forty-two of its patents. *See infra* note 164.

89. Steven Shankland, *Open-source Allies Go on Patent Offensive*, CNET NEWS.COM, Aug. 11, 2005, http://www.news.com/2100-7344_3-5827844.html (last visited Mar. 7, 2008).

90. *Id.*

91. GPLv2, *supra* note 5, at pmbl.

92. *Id.* § 7.

93. *Id.*

downstream recipients, including those that do not receive the software directly from the patent holder, or to simply refrain from distributing the patented software.[94] Stallman calls this concept "Liberty or Death."[95] One interpretation of section 7 is that it creates an implied royalty-free patent license.[96] Based on this interpretation, Stallman and the FSF maintain that GPLv2 contains an implied patent license based on U.S. copyright law.[97] Even free software supporters, however, admit that the scope of such an implied license would be difficult to determine.[98]

C. GPLv3 Goals and Drafting Process

Richard Stallman and Eben Moglen identified several issues to tackle for GPLv3.[99] First, they wanted to make GPLv3 more internationally compliant, as GPLv2 relied on key terms in U.S. copyright law, such as derivative works and collective works.[100] Second, the FSF wanted GPLv3 to continue to be a standard for open source licenses.[101] Third, the FSF wanted to adapt the license to legal and technological developments since

94. *See id.* at pmbl. ("[W]e have made it clear that any patent must be licensed for everyone's free use or not licensed at all.); *Id.* § 7 ("If you cannot distribute as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all.").

95. Richard Stallman, Address at the 2nd International GPLv3 Conference (April 21, 2006), http://fsfeurope.org/projects/gplv3/fisl-rms-transcript.en.html (last visited Apr. 4, 2008).

96. ROSEN, *supra* note 22, at 126 ("[Under section 7], a licensor cannot distribute software . . . while simultaneously demanding royalties for his patents. His act of distributing the software implies a royalty-free license."). Other commentators have also raised the possibility of implied licenses in GPLv2. *See* Adam Pugh & Laura A. Majerus, Potential Defenses of Implied Patent Licenses Under the GPL, http://www.fenwick.com/docstore/Publications/IP/potential_defenses.pdf (last visited Apr. 4, 2008) (discussing an implied license based on theories of legal estoppels, equitable estoppels, conduct, and acquiescence).

97. *See* Stallman, *supra* note 95. Stallman stated:

> [I]n GPL version two we rely on an implicit patent licence. The issue is: what if the company who distributed the GPL covered program has a patent that applies to something in the program? . . . In the US, by distributing the program to you under the GPL they're saying they have no objectins [sic] if you carry out your rights under the GPL, so if they then try to sue you for doing so, they will lose.

Id.

98. SOFTWARE FREEDOM LAW CENTER, A LEGAL ISSUES PRIMER FOR OPEN SOURCE AND FREE SOFTWARE PROJECTS 25 (2008), http://www.softwarefreedom.org/resources/2008/foss-primer.pdf (last visited Apr. 4, 2008).

99. Moglen, *supra* note 21.

100. *Id.* For more discussion, *see supra* Section III.A.

101. Moglen, *supra* note 21.

the release of GPLv2, including the emergence of software patents, the anti-circumvention provisions of the Digital Millennium Copyright Act ("DMCA"), and Digital Rights Management technology.[102] In FSF's view, these developments placed restrictions and burdens on users and developers of free software.[103]

Nonetheless, the FSF recognized the commercial interests involved in open source, and invited those interests to participate in the drafting process.[104] After eighteen months of revision, including comment by several different discussion committees representing various open source interests on four different discussion drafts of the license, the final version of GPLv3 was released on June 29, 2007.[105]

IV. ANALYSIS OF THE MAIN PROVISIONS OF GPL VERSION 3

This Part analyzes the major changes in GPLv3, which attempt to address the legal problems in GPLv2, and whether the updated license successfully resolves these concerns.[106] Section IV.A reviews new language in GPLv3 intended to clarify the license scope and licensors' obligations.[107] Section IV.A.1 looks at GPLv3's explicit references to dynamic linking,[108] while Section IV.A.2 examines the consequences of GPLv3 for kernel modules,[109] which are not specifically addressed by the new version. GPLv3 enacts an express patent license, which is examined in detail in Section IV.A.3. Finally, Section IV.B examines other GPLv3 provisions meant to modernize the license and deal with what the FSF viewed as increasing threats to software freedom. Given the license scope ambiguities as well as the difficulties classifying dynamic linking and Linux kernel modules under GPLv2, many hoped that GPLv3 would provide more clarity. While the new license unifies and clarifies the license language, it still contains ambiguities.

102. *See* FREE SOFTWARE FOUNDATION, GPLV3 FIRST DISCUSSION DRAFT RATIONALE 3 (2007), http://gplv3.fsf.org/gpl-rationale-2006-01-16.pdf (last visited Apr. 4, 2008).

103. *See id.*

104. FREE SOFTWARE FOUNDATION, GPL3 PROCESS DEFINITION 9 (2006), http://gplv3.fsf.org/gpl3-process.pdf (last visited Apr. 4, 2008).

105. Gomulkiewicz, *supra* note 3, at 15.

106. This Note does not attempt to discuss exhaustively every one of GPLv3's changes, but instead focuses on the most significant provisions.

107. For a discussion on GPLv2 license scope, *see supra* Section III.A.

108. For a discussion of the dynamic linking problem, *see supra* Section III.A.1.

109. For a discussion of problems related to Linux kernel modules, *see supra* Section III.A.2.

A. GPLv3's Clarifications of License Scope and Copyleft Obligations

GPLv3 modifies the language of the license scope and copyleft obligations while attempting to maintain the same copyleft spirit of its predecessor: a licensee is permitted to freely copy, redistribute, and improve software governed by the license, provided that the licensee makes the source code, either original or modified, available to downstream recipients: "You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply . . . to the whole of the work, and all its parts . . ."[110]

GPLv3 employs newly defined terms, such as "propagate" and "convey" in describing the potential distribution of source code.[111] The new definitions also make clear that to "modify" a work is to create a "modified version" of the earlier work or a work "based on" the earlier version, allowing for a consistent terminology in the license instead of GPLv2's inconsistent use of the terms "modified work" and "work based on the

110. GPLv3, *supra* note 5, § 5(c).

111. GPLv3, *supra* note 5, § 0. To "propagate" a work is to:

> . . . do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

Id. To "convey" a work is to engage in "any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying." *Id.* These changes arguably eliminate GPLv2's dependence on U.S. copyright terms, such as "derivative work" or "distribute." At the same time, this means that lawyers can no longer rely on U.S. case law and statutes to provide the meanings of the license terms. Eben Mogen, at a talk describing GPLv3, stated:

> I will say, therefore, the second discussion draft of GPL3 will not have any longer any dependency on the American copyright law idea of a "derivative work." Accordingly we will be able to avoid another fifteen years of complaining from lawyers who quite justifiably thought the reliance on a US centric view of the derivative work was a major source of uncertainty and uneasiness. They may think that what we have done instead is not a perfect improvement, but I feel certain that there will at least be the recognition that it's a useful development.

Eben Moglen, Address at the 3rd International GPLv3 Conference (June 22, 2006), http://www.fsfeurope.org/projects/gplv3/barcelona-moglen-transcript.en.html (last visited Apr. 4, 2008).

Program."[112] Abandoning GPLv2's use of term "derivative work," GPLv3 defines a "covered work" to mean either the "unmodified Program or a work based on the Program."[113] These definitional changes clean up the GPL and create more uniformity in the license.

The new license consolidates the scope of the copyleft obligation to provide source code in section 1.[114] GPLv3 distributors must include all source code needed to generate, install, and modify the work.[115] However, the scope of the obligation to provide source code exempts system libraries,[116] which are program subcomponents included in either (1) the core components of a compiler or an operating system (such as a kernel)[117] or (2) the implementation of a standard interface.[118]

These provisions attempt to clarify the "special exception" in GPLv2 for system libraries.[119] The FSF claims the new GPLv3 provisions extend the system library exception to include software that might not come directly with the operating system (such as software libraries of popular programming languages like Java) and make clear that distributors can combine GPLv3 software with non-GPL system libraries.[120]

112. GPLv3, *supra* note 5, § 0, para. 4. *See supra* notes 46 & 57 and accompanying text.

113. GPLv3, *supra* note 5, § 0, para. 5. GPLv2's use of "derivative work" raised potential ambiguities in the license. For a full discussion, *see supra* Section III.A.

114. GPLv3, *supra* note 5, § 1 (defining "Corresponding Source" to mean "all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work.").

115. *Id.*

116. *Id.* § 1, para. 4. GPLv3 defines "System Libraries" as:

> . . . anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, *and* (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form.

Id. (emphasis added).

117. *Id.* § 1, para. 3 (defining "Major Component" as "a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.").

118. *Id.* § 1, para. 2 (defining "Standard Interface" as "an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.")

119. GPLv2, *supra* note 5, § 3, para. 2. The "special exception" in GPLv2 is discussed in Section 0 of the license. *See supra* Section III.A.1.

120. BRETT SMITH, A QUICK GUIDE TO GPLV3 5 (2007), http://www.gnu.org/licenses/quick-guide-gplv3.pdf (last visited Apr. 4, 2008).

1. *Clarification of Dynamic Linking*

GPLv3 includes an express provision by the FSF to address the dynamic linking issue that plagued GPLv2.[121] The scope of the copyleft obligation to provide source code for a GPLv3 work now extends to dynamically linked subprograms which share "intimate data communication or control flow" with the work.[122] While attempting to clarify dynamic linking, this language actually muddles the license scope. The problem is that the FSF strikes a tenuous position, stating first that dynamically linked programs are within the scope of the source code provision, then pulling back with a cryptic condition that the dynamic links must involve "intimate data communication or control flow." Exactly what is considered "intimate" enough for this provision is unclear. This change appears to do little more than codify the FSF's "unofficial" position on GPLv2 dynamic linking into GPLv3.[123] At the very least, however, users, developers, and distributors can no longer advocate a categorical exclusion of dynamic linking from the scope of the copyleft obligation under GPLv3, as some did under the previous license.[124] But the new language leaves the status of any given dynamically linked program unclear, perhaps awaiting the development of case law interpreting "intimate."

2. *Linux Kernel Modules*

GPLv3 does not expressly address the effect of combining non-GPL kernel modules with GPLv3 code. Thus, it is unclear whether the "intimate data communication" provision discussed above changes the existing

121. This GPLv3 provision is contained in section 6, see *infra* note 122 and accompanying notes. The dynamic linking issue in GPLv2 is discussed in Section III.A.1. *See supra* Section III.A.1.

122. Conveying a GPLv3 program in object code form requires the conveyer to supply the "Corresponding Source." GPLv3, *supra* note 5, § 6. "Corresponding Source" is defined as "all source code needed to generate, install, and (for an executable work) run the object code and to modify the work." *Id.* § 1, para. 4. Corresponding Source also includes "interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, *such as by intimate data communication or control flow between these subprograms and other parts of the work*." *Id.* (emphasis added).

123. *See Mere Aggregation*, *supra* note 70 and accompanying notes. The FSF always maintained the position that certain types of dynamic linking create derivative works subject to the terms of the GPLv2. *Id.*

124. *See* Determann, *supra* note 39, at 1458-62, 1488 n.255 (stating "dynamic linking" of libraries "would not qualify as derivative works"); Stoltz, *supra* note 39, at 1458-59, 1463-64 (discussing the "linking exemption" to the derivative work right and how it applies to GPLv2).

treatment of Linux kernel modules.[125] One interpretation is that the FSF, by codifying its "unofficial" position on GPLv2 dynamic linking into GPLv3,[126] also codified its position that proprietary kernel modules that dynamically link to the Linux kernel are subject to the copyleft obligation of the GPL.[127]

Another approach is to examine how Linux kernel modules interact with the Linux kernel, as well as the amount of "intimate data communication and control flow" involved in this process. The Linux kernel interacts with kernel modules using a specific software interface.[128] Depending on the kernel module's function and design, a module might have a simple or complex interface to the Linux kernel.[129] The complexity of the interface is at least partly a function of the complexity of the underlying module: is it a module that simply prints to the screen, or is it a network card driver that passes huge amounts of Internet data to the user?[130] The more complex the interface, and the more data exchanged between the module and the kernel, the more likely there will be "intimate data communication and control flow" under the terms of GPLv3. The appropriate paradigm is a spectrum, where on one end there is minimal data sharing and a simple interface, and on the other there is "heavy" data sharing and a complex interface, with GPLv3's undefined definition of "intimate" lying somewhere in between. Companies with proprietary kernel modules must therefore be careful about where their module falls on this spectrum.

3. Impact Regarding Scope, Dynamic Linking and Kernel Modules

Regarding (1) license scope, (2) dynamic linking, and (3) Linux kernel modules, GPLv3 introduces new terms that try to clarify the scope of the copyleft obligation by unifying language and explicitly stating that certain dynamically linked programs are covered under the license. Unfortunately, the inclusion of these new terms and conditions creates new uncertainties for companies seeking a clear delineation of their potential liability under GPLv3. While the GPLv2 had its own problems, there was a level of comfort in the computing and legal industry regarding plausible interpretations

125. The "intimate data sharing" provision was added in GPLv3 to address dynamic linking. *See supra* Section IV.A.1.

126. *See supra* note 123 and accompanying text. The FSF always maintained the position that certain types of dynamic linking create derivative works subject to the terms of the GPLv2. *Id.*

127. *See supra* note 80 & 81 and accompanying text.

128. RUBINI, ET AL., *supra* note 74, at 1, 5-6; Hass, *supra* note 20, at 251, 253.

129. *See* Hass, *supra* note 20, at 254.

130. *See id.* at 254.

of the license.[131] GPLv3 introduces new language in an attempt to clarify a licensor's obligations, but new language inevitably introduces new ambiguities.

For proprietary software vendors, GPLv3 is a mixed bag. Clarifications in the scope of the copyleft obligation will allow vendors to better plan how to deal with GPLv3 software and ensure the closed-source integrity of their proprietary code. However, the addition of language covering dynamic linking could compromise a vendor's proprietary kernel modules. The new language could be read to extend GPLv3's copyleft obligation to certain proprietary kernel modules, which should be a concern for businesses dependent on kernel module functionality.

The FOSS community stands to gain from the additional clarification in the license. The clarifications in the license scope allow FOSS programmers to better grasp their obligations under the license. At the same time, the FOSS community is split over whether or not the GPL copyleft obligation extends to dynamically linked kernel modules. FSF and Stallman, the authors of GPLv3, added new language to cover certain types of dynamic linking.[132] Linus Torvalds, who accepts the existence of proprietary kernel modules, will probably continue to maintain that the new language does not cover Linux kernel modules.[133]

B. Software Patent Problems

GPLv3 continues GPLv2's aversion to software patents,[134] but does not stop at "Liberty or Death."[135] The new version adds four significant

131. *See, e.g.*, Steven Shankland, *Sun Considers GPL for Solaris*, CNET NEWS.COM, Nov. 14, 2006, http://www.news.com/Sun-considers-GPL-for-Solaris/2100-7344_3-6135461.html (last visited Apr. 4, 2008) (Sun's Executive Vice President of Software, Rich Green, stating there is a "familiarity and comfort level with" GPLv2); Interview with Heather J. Meeker, Shareholder, Greenberg Traurig, in Palo Alto, Cal. (Oct. 9, 2007) (indicating a general level of comfort and familiarity with GPLv2 built up over its sixteen year existence).

132. *See supra*, note 122 and accompanying text.

133. *See supra*, note 79.

134. GPLv3's preamble reads: "States should not allow patents to restrict development and use of software . . . ." GPLv3, *supra* note 5, at pmbl.

135. GPLv3 also contains a "Liberty or Death" provision in section 12, which states:

> If conditions are imposed on you (whether by court order, agreement, or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all.

changes to the way the GPL deals with patents: (1) a patent retaliation clause;[136] (2) an explicit patent license for contributors;[137] (3) a "knowing reliance" provision to shield downstream users;[138] and (4) two ad hoc clauses dealing with an ongoing collaboration deal between Microsoft and Novell.[139]

1. Patent Retaliation Provision

Unlike GPLv2, GPLv3 contains a patent retaliation provision,[140] a concept borrowed from other open source licenses, such as the Apache License and Mozilla Public License. Under a typical open source patent retaliation clause, a licensee who sues a licensor for patent infringement on claims covering the licensed software would have its license terminated; in other words, a licensee cannot keep its license while simultaneously filing a patent infringement suit.[141] GPLv3's patent retaliation provision states that:

> You [the licensee] may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other

Id. § 12. Note that this new provision is not limited just to obligations or conditions arising from patents, but to any condition which "would contradict the conditions of this License." *Id.*

136. *Id.* § 10, para. 3.

137. *Id.* § 11, para. 2-4.

138. *Id.* § 11, para. 5.

139. *Id.* § 11, para. 6-7.

140. *Id.* § 10, para. 3.

141. *See, e.g.*, Apache License, *supra* note 33, § 3. The Apache License provides that:

> If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

Id. See also Mozilla Public License, *supra* note 33, § 8.2. The Mozilla Public License states that:

> If You initiate litigation by asserting a patent infringement claim (excluding declatory [sic] judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that . . . such Participant's Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate . . . .

Id.

> charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.[142]

If a licensee proceeds to file patent infringement litigation against the licensor, section 8 will immediately terminate its license.[143] GPLv3's patent retaliation clause follows the typical approach: the licensee cannot assert any patent claim on the licensed program against the licensor without termination.[144]

There are two potential problems with the patent retaliation provision. First, the scope of this provision is uncertain because GPLv3 defines "the Program" as "*any* copyrightable work licensed under this License," with "this License" defined as GPLv3.[145] The way that the "any" is placed in the definition of "the Program" above could be interpreted to mean that the patent retaliation provision would cover not just the specific program licensed under GPLv3, but all GPLv3-licensed programs. However, this is an extreme interpretation that appears to result from bad license drafting, so a court is unlikely to take a similar approach. The alternative, and more likely interpretation, is that "the Program" refers just to the specific program licensed under GPLv3. In fact, the FSF published a small note on its website discounting the extreme interpretation and endorsing the alternative interpretation.[146]

Second, nothing in a patent retaliation clause prevents a licensee from immediately terminating distribution of the licensed work and proceeding to sue the licensor. Thus, this deterrent may not be effective against motivated licensees. However, it may not be feasible for many licensees to immediately terminate use or distribution of the licensed work, especially if it is embedded in the licensee's own product. Furthermore, if the licensee never used or distributed the licensed work, it probably did not need a license to begin with. Despite the first scenario, GPLv3's patent retaliation provision will be an effective deterrent in most typical instances.

142. GPLv3, *supra* note 5, § 10, para. 3.

143. *See id.* § 8 (stating that "You may not propagate or modify a covered work except as *expressly provided* under this License. Any attempt *otherwise* to propagate or modify it *is void*, and will *automatically terminate your rights* under this License . . . .") (emphasis added).

144. *See id.* § 10, para. 3.

145. *Id.* § 0 (emphasis added).

146. *See* Free Software Foundation, What Does "the Program" Mean in GPLv3?, http://www.gnu.org/licenses/gplv3-the-program.html (last visited Apr. 4, 2008).

2. *Express Patent License*

GPLv3 contains an express patent license.[147] If a licensee modifies GPLv3 code and distributes the modified version as a *contributor*, then the licensee grants an express, nonexclusive, worldwide, royalty-free patent license to all of the contributor's "essential patent claims" in the entire modified version *as a whole*.[148] This patent license raises three main issues.

First, the patent license applies only to contributors of modified GPLv3-licensed code, and not mere distributors of unmodified code.[149] Although earlier drafts had included patent licenses covering both distributors and contributors, the FSF removed the patent license for distributors from the final draft because of pressure from commercial distributors of open source software that had strategic patent portfolios.[150]

Second, GPLv3's patent license is broader than most other open source licenses, which typically license patent claims reading solely on the contribution or a combination of that contribution with the remainder of the work, and not the entire work (with the embedded contribution) *as a whole*.[151] Under GPLv3, contributors might grant patent licenses reading on code they never modified or wrote. The FSF decided to keep a broad patent license in GPLv3 despite concerns from some companies that wanted the more common open source approach, which only granted patent licenses on "changes" and "additions" a contributor made to the work.[152]

Third, and most crucial, is delineating the scope of the patent license. GPLv3 defines "essential patent claims" as:

> [A]ll patent claims owned or controlled by the contributor, whether *already acquired or hereafter acquired*, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, *but do not include claims*

147. The FSF always maintained the controversial position that GPLv2 contained an implied patent license. *See supra* Section III.B.

148. GPLv3, *supra* note 5, §§ 5(c), 11.

149. *See* GPLv3, *supra* note 5, § 11 (defining a "contributor" as a "copyright holder who authorizes use under [GPLv3] of the Program or a work on which the Program is based").

150. *See* FREE SOFTWARE FOUNDATION, GPLV3 THIRD DISCUSSION DRAFT RATIONALE 15 (2007), http://gplv3.fsf.org/gpl3-dd3-rationale.pdf (last visited Apr. 4, 2008).

151. *Compare* GPLv3, *supra* note 5, §§ 5(c), 11, *with* Apache License, *supra* note 33, § 3, *with* Mozilla Public License, *supra* note 33, § 2.2(b).

152. THIRD DISCUSSION DRAFT RATIONALE, *supra* note 150, at 18.

> *that would be infringed only as a consequence of further modification of the contributor version.*[153]

A contributor would certainly grant a license for patent claims reading on the contributions that it has made. Further downstream recipients of the contributed code also receive this patent license. The question, however, is whether the contributor's patent license can be enlarged based on further modifications of its contribution by downstream recipients. According to the FSF, the italicized limitation in the definition means that "the set of essential patent claims . . . is fixed by the particular *version* of the work that was contributed."[154] Thus, further modifications to the contribution by downstream recipients cannot enlarge the scope of the contributor's patent license.[155] The scope of such derivative works is potentially unbounded, and therefore would be of concern to any business with an important patent portfolio who contributed code. Despite the change, an ambiguity remains in the language. The FSF uses the word "version" to describe the point at which the patent license is fixed.[156] This word is ambiguous: is the "version" fixed at the time at which the contribution is made? As versions in software programs might only change every few months, can multiple contributions be made under the same "version"?

Additionally, the patent license is temporally unbounded, as "essential patent claims" includes all of the contributor's patent claims "already acquired or *hereafter acquired*."[157] However, this infinite capture period appears to conflict with FSF's claim that a contributor's patent license is fixed at the time of contribution. There are also potential ambiguities regarding how far this period can extend. First, does the infinite capture period cover patents that a contributor might later acquire by assignment or acquisition? For example, if company X contributes GPLv3-licensed code, and later decides to purchase target company Y with a significant patent portfolio, does the patent license granted when X contributed the code include claims from company Y's patents? Second, does the phrase "owned or controlled by the contributor" reach down to include patents owned by subsidiaries or affiliates? For example, if company X spins out company Z as a wholly owned subsidiary, assigning all its patents to Z with an exclusive grant-back license to those patents, and then contributes GPLv3-license code, does X grant a patent license under the GPLv3? The license language is silent on how it would handle these scenarios.

153. GPLv3, *supra* note 5, § 11, para. 2 (emphasis added).
154. THIRD DISCUSSION DRAFT RATIONALE, *supra* note 150, at 20 (emphasis added).
155. *See id.*
156. *Id.*
157. GPLv3, *supra* note 5, § 11, para. 2 (emphasis added).

Companies with important patent portfolios should be wary of GPLv3's patent license, especially if they employ open source project contributors. Contributor companies should employ a careful review policy to ensure contributions do not cause unwitting patent licenses.

3. *"Knowing Reliance" and Downstream Users*

GPLv3 attempts to shield downstream licensees of a program from third party patent claims.[158] This "knowing reliance" provision provides that if a licensor knows that the GPLv3 work he is conveying is covered by a third party patent, and licenses the patent from that third party in order to use the work, the licensor cannot convey the work to downstream users because the downstream users—although under the GPL they would receive copyright rights and licenses to patents owned by the licensor—would not be licensees of the third-party patent holder and would become infringers of the third-party's patents.[159] For example, suppose company A wishes to distribute a program it developed under GPLv3 to company B. Company A also knows that company Z's patent covers this program and has obtained a license from Z in order to make, use, and distribute the program. However, company Z's patent license does not extend to company B or any subsequent downstream recipients of company A's program. Therefore, company A knows that B's use of the distributed program would infringe on Z's patent. In this situation, company A (the licensor) would be unable to distribute the program because it would violate the "knowing reliance" provision in GPLv3. In sum, GPLv3 mandates that licensors cannot "knowingly rely" on patent licenses to protect themselves while exposing their downstream recipients to patent infringement.

However, GPLv3 gives a licensor three "escape" options to allow for distribution of the work in the situation above: (1) cause the source code of the work to be made available for anyone to copy, free of charge and under the terms of GPLv3, through a public network server; (2) deprive itself of the benefit of the patent license for this particular work; or (3) arrange to extend the patent license to downstream recipients via a sublicense.[160] Most companies will probably choose option (1), as option (2) is a harsh penalty that might have consequences for a licensor's business, and option (3) is usually not available as the vast majority of commercial patent cross-licenses do not allow for sublicensing. Note that option (1) requires that source code of the appropriate work be made available to *anyone* via the Internet, which is a broader obligation than the normal

158. *See id.* § 11, para. 5.
159. *Id.*
160. *Id.*

GPLv3 requirement that source code made available only to recipients of conveyed object code.[161]

The definition of "knowing reliance" also raises another issue. GPLv3 defines "knowing reliance" as "actual knowledge that, but for the [third party] patent license, your [the licensor] conveying the covered work . . . or your recipient's use of the covered work in a country, would infringe one or more identifiable patents . . . *you have reason to believe are valid*."[162] This would indicate that if the licensor believed that the third party patents were not valid, he would not be in violation of the "knowing reliance" provision by conveying the "covered work." This could have significance for companies who license third party patents they believe are invalid because it is cheaper to take a license to the patent instead of being embroiled in patent litigation.[163] The italicized proviso above appears to open the door for such companies to escape the knowing reliance provision.

4. *Addressing the Microsoft/Novell Agreement*

In November 2006, during the drafting of GPLv3, Microsoft arranged a controversial deal with Linux distributor Novell.[164] In that deal, Microsoft extended a limited covenant not to assert its patents against Novell Linux customers in exchange for money and other considerations from Novell.[165] Microsoft and Novell structured the deal with the creative solution of a covenant not to sue instead of a patent royalty, as GPLv2's "Liberty or Death" clause prevents Novell from paying any royalty to Microsoft to distribute Linux.[166] Members of the free software movement, including Stallman, were outraged that Novell had entered into such a deal,

161. *Compare* GPLv3, *supra* note 5, § 11, para. 3, *with* GPLv3, *supra* note 5, § 6.

162. GPLv3, *supra* note 5, § 11, para. 5 (emphasis added).

163. Major technology companies often license what they believe to be meritless patents from so-called "patent trolls" because of the inherent uncertainty in patent litigation. Joe Beyers, *Perspective: Rise of the Patent Trolls*, CNETNEWS.COM, Oct. 2, 2005, http://www.news.com/Rise-of-the-patent-trolls/2010-1071_3-5892996.html (last visited Apr. 4, 2008) (discussing the rise of patent trolls that extract money from companies due to uncertainty in patent infringement suits). "Patent troll" is a pejorative term that is used to describe a company whose primary business is to obtain and enforce patents, and not release products. *See* Steve Seidenberg, *Troll Control*, 92 A.B.A. J. 50 (2006) (discussing the definition of patent trolls).

164. Roger Parloff, *Microsoft Takes on the Free World*, FORTUNE, May 28, 2007, at 76.

165. *Id.* Microsoft claims that free and open source software infringes on 235 of its patents. Specifically, it claims the Linux kernel infringes on forty-two of the 235 patents. *Id.*

166. *Id. See also* GPLv2, *supra* note 5, § 7, para. 1.

as they believed these arrangements could render free software effectively proprietary.[167] Stallman and Moglen immediately set out to include provisions in GPLv3 to prevent a similar deal in the future.[168]

The FSF's answer to the Microsoft-Novell deal was two additional GPLv3 provisions. The first provision, aimed at Microsoft, states that where a patent holder distributes GPLv3 code but only grants a patent license to some subset of parties receiving the covered work, the license is automatically extended to all recipients of the covered work.[169] If Microsoft is bound under the provision, Microsoft's patent license to Novell Linux customers would extend to all Linux users. However, the enforceability of this provision on Microsoft is doubtful, as Microsoft is not a party to GPLv3 and does not distribute GPLv3 code.[170]

The second provision, aimed at Novell, forbids distributors of GPLv3 code from entering into "discriminatory patent licenses" with third parties that are "in the business of distributing software."[171] A "discriminatory patent license" is one which does not include within its scope all of the "rights that are specifically granted under [GPLv3]."[172] This provision specifically grandfathers Novell's agreement with Microsoft, and all such "discriminatory patent license" agreements made before March 28, 2007.[173] The limitation of this provision to third parties "in the business of distributing software" appears allow distributors to enter into "discriminatory patent licenses" with businesses that do not distribute software, such

167. Parloff, *supra* note 164. The article notes:

> FOSS developers, who do not have the resources to defend themselves against a Microsoft patent suit, felt safe as long as powerful corporate Linux users shared their cause. But now the big boys could just buy their Linux from a royalty-paying vendor like Novell, getting protection from lawsuits and leaving the little guys to fend for themselves. What the shortsighted corporate types didn't grasp was that without the little-guy developers there might not be any high-quality FOSS for them to use five years down the road.

Id. See also THIRD DISCUSSION DRAFT RATIONALE, *supra* note 150, at 25.

168. Parloff, *supra* note 164.

169. GPLv3, *supra* note 5, § 11, para. 6.

170. Microsoft released a statement after GPLv3 was published, taking pains to state that it was not a party to GPLv3, and that its deal with Novell would remain valid even if Novell started distributing GPLv3 code. *See* Microsoft's Statement on GPLv3, http://www.microsoft.com/about/legal/intellectualproperty/GPLv3.mspx (last visited Apr. 28, 2008).

171. GPLv3, *supra* note 5, § 11, para. 7.

172. *Id.*

173. *Id.*

as patent trolls and other similar entities.[174] While these two provisions may prevent or deter a future agreement similar to Microsoft-Novell, they encumber an already complex license with two overly inclusive paragraphs.

C. A Survey of Other GPLv3 Provisions

Although patents and license scope were two of the biggest changes in GPLv3, the new license also contains important updates meant to deal with advancements in technology since the release of GPLv2 in 1991. The most important and controversial changes are analyzed below.

1. Modification of Software for Consumer Products

The FSF has been openly hostile to a practice dubbed "Tivoization."[175] Tivoization is the practice of incorporating GPL software on a consumer device or appliance that the user cannot change, because the manufacturer uses hardware to disable the device if it detects modified software.[176] The FSF believes that these device manufacturers take advantage of the freedoms the GPL provides, but do not let users do likewise.[177] The term originates from the TiVo, a company that utilized GPLv2 licensed software on its digital video recorder products, but prevented users from loading modified versions of the TiVo software on its devices.[178] GPLv3 requires that distributors provide information necessary "to install and execute modified versions of a covered work" in their consumer products, thereby allowing users to run modified code on their consumer devices.[179]

However, the license contains two important exceptions to this requirement. The first exception is for devices where the software can never be modified (e.g., if it is installed in read-only memory, or ROM).[180] This exception will likely be unpopular because restriction to a ROM is a serious technical limitation: post-distribution patches and upgrades cannot be

174. "Patent troll" is a pejorative term that is used to describe a company whose primary business is to obtain and enforce patents, and not release products. *See* Seidenberg, *supra* note 163, at 50.

175. *See* Stallman, *supra* note 39. GPLv3's preamble states this hostility succinctly: "Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software." GPLv3, *supra* note 5, at pmbl.

176. Gomulkiewicz, *supra* note 3, at 16.

177. Stallman, *supra* note 39.

178. Gomulkiewicz, *supra* note 3, at 16.

179. GPLv3, *supra* note 5, § 6, para. 4.

180. *Id.*

applied as the contents of a ROM are fixed.[181] The second exception is that a user's device can be denied access to the network if the modified code "materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network."[182] This definition is potentially ambiguous, and companies might try to exploit this by taking a broad interpretation of what violates the "rules and protocols" for network communication to deny access. Note that this exception only allows such companies to deny users access to the network, and not to the device itself, although many modern consumer products are networked.

2. *Digital Rights Management ("DRM")*

The FSF opposes the use of DRM[183] and other similar digital controls on ideological grounds.[184] Although GPLv3 does not ban the use of DRM in GPLv3 code, the license allows developers to subsequently remove DRM from GPLv3 code without running afoul of so-called "anti-circumvention laws" that prohibit tampering with DRM technology, such as the U.S. Digital Millennium Copyright Act ("DMCA").[185] First, section 3 states that a work licensed under GPLv3 does not qualify as an "effective technological measure" under the DMCA.[186] Second, GPLv3 contains an explicit waiver by GPLv3 licensors of the protections against DRM circumvention in the DMCA.[187] Courts may be willing to enforce such a waiver against a software distributor who is a voluntary party to the GPLv3 license, but a third party content owner or distributor who is not a party to the license may still be able to invoke the DMCA against users of GPLv3 code engaging in circumvention.

181. *See* HENNESSEY & PATTERSON, *supra* note 9, at B-13 to 14.

182. *See* GPLv3, *supra* note 5, § 6, para. 6.

183. *See* Stallman, Opposing Digital Rights Management, http://www.gnu.org/philosophy/opposing-drm.html (last visited Apr. 4, 2008). DRM refers to the panoply of technologies designed to allow copyright holders to limit access and usage of digital media or devices. *Id.*

184. *See, e.g.*, Stallman, *supra* note 39; FIRST DISCUSSION DRAFT RATIONALE, *supra* note 102, at 3.

185. *See* 17 U.S.C. § 1201 (2000). The DMCA states that "[n]o person shall *circumvent a technological measure* that effectively controls access to a work protected under [this Act]." 17 U.S.C. § 1201(a)(1)(A) (2000) (emphasis added).

186. GPLv3, *supra* note 5, § 3, para. 1.

187. *Id.* para. 2.

3. *Termination and Cure*

GPLv2 terminated automatically upon failure to comply with its terms, meaning that continued use of the program was copyright infringement.[188] GPLv2 did not address how to reinstate rights under the license. The FSF believed this policy was too harsh, as it punished even inadvertent violators.[189] GPLv3 addresses this problem by providing an express cure period for violators of the license.[190] Specifically, the license states:

> If you [the licensee] cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.[191]

These cure provisions allow inadvertent violators to retain a provision license immediately after they come back into compliance, and permanently if the violator is not notified by the copyright holder sixty days after achieving compliance.

V. THE PRESENT AND FUTURE OF GPL VERSION 3

This Part examines additional issues confronting GPLv3 as it moves into the future. Section V.A examines the FOSS contribution problem created by incompatible licenses and license versions. Section V.B looks at the adoption of GPLv3 to date, while Section V.C assesses the future of GPLv3.

A. The FOSS Contribution Problem

Although GPLv3 maintains the same copyleft spirit of its predecessor, GPLv3 is not compatible with GPLv2.[192] This incompatibility means that GPLv2-licensed source code cannot be combined with source code li-

188. *See* FIRST DISCUSSION DRAFT RATIONALE, *supra* note 102, at 17.
189. *Id.*
190. GPLv3, *supra* note 5, § 8, para. 2.
191. *Id.*
192. *See* Stallman, *supra* note 39. Stallman has stated:

> GPL version 2 will remain a valid license, and no disaster will happen if some programs remain under GPLv2 while others advance to GPLv3. These two licenses are incompatible, but that isn't a serious problem. When we say that GPLv2 and GPLv3 are incompatible, it means there is no legal way to combine code under GPLv2 with code under GPLv3 in a single program.

Id.

censed under GPLv3.[193] Code licensed under GPLv2, in many cases, must be relicensed under GPLv3 to be compatible, which can be a significant obstacle for large FOSS projects.

The person or entity wishing to re-license source code under an incompatible license must own the copyright to the work.[194] Having a single entity own all the copyrights in the source code of an FOSS project is inherently difficult because the code may originate from dozens, if not hundreds or thousands of developers.[195] Each of these developers owns the copyright in its individual contribution, which is licensed under the project's open source license.[196] The project will have difficulty relicensing its source code unless the developers assign their copyrights or give express permissions at the time of their contribution to a single entity.[197] Otherwise, the project likely must go back to obtain permission from each individual contributor.[198]

The FSF has recognized this problem with GPLv2 and came up with two solutions. The first solution was to start using contributor agreements with FSF-owned GNU projects.[199] The second solution was to encourage projects to license under a specific version or "any later version," thus allowing existing projects licensed under GPLv2 or "any later version" to be re-licensed under GPLv3 without direct contributor approval.[200]

B. FOSS Adoption of GPLv3

As of April 2008, a prominent database tracking GPLv3 adoption indicated that over 2000 FOSS projects out of nearly 8500 in the database have converted to GPLv3.[201] Some important and visible projects that

193. Incompatibility frustrates source code sharing, one of the fundamental benefits of the FOSS development model.

194. *See* ROSEN, *supra* note 22, at 47 ("License compatibility is not an issue for projects that are copyright and patent owners, because the contributors no longer have any right to refuse the projects' licensing decisions for contributions the contributors no longer own.").

195. *See* Mann, *supra* note 4, 14-15, 15 n.59, 27-28 (describing the operation of open source software licenses).

196. *See id.* at 14-15.

197. *See id.*

198. *See id.* at 15 n.59.

199. Copyright Assignment, http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html#AssignCopyright (last visited Apr. 4, 2008).

200. SMITH, *supra* note 120, at 4.

201. Palamida GPLv3 and LGPLv3 Information Site, http://gpl3.palamida.com (last visited Apr. 4, 2008). Note that this database is driven mainly by user submissions, so the total number of projects reported does not nearly match the number of total projects li-

have already converted to GPLv3 include Samba[202] and SugarCRM.[203] Sun had indicated that it would seriously consider re-licensing OpenSolaris under GPLv3,[204] but there has been no official announcement by the company indicating such a move after the release of GPLv3. However, Sun did release its first application licensed under GPLv3, characterizing it as a "first step" and suggesting that future products could be released under the license.[205]

C. Envisioning the Future of GPLv3

The GPLv3 has attempted to update a nearly sixteen-year-old GPLv2 and clarify its ambiguities. However, as outlined in Part IV, there are many legal issues that still remain unresolved with the new license, such as the treatment of dynamic linking and propriety kernel modules. Furthermore, the license contains many additions which are motivated by the philosophical views of Richard Stallman and the FSF, including provisions dealing with DRM, Tivoization, an express patent license, and the Microsoft/Novell agreement. Those additions will inevitably add complexities and legal uncertainties to GPLv3.

Given this backdrop, free software and open source projects and companies will likely adopt the license for both philosophical and practical reasons. Projects might adopt GPLv3 because they identify with the ideology of the FSF and want to encourage use and adoption of the new license. Adopting GPLv3 can also publicize a company's open source efforts and incur political goodwill from the FOSS community. Adoption of GPLv3 will likely occur not because it is a better legal document than GPLv2, but because it is the newest version of the most well known FOSS license today. New open source projects and smaller projects might decide to move to new license, but Linux and other major projects, especially if they are backed by commercial interests, will not do so immediately be-

censed under GPLv3 as reported by SourceForge.net (many of which are probably inactive). For SourceForge statistics, see SourceForge, *supra* note 34 and accompanying text.

202. Samba, Samba Adopts GPLv3 for Future Releases, http://news.samba.org/announcements/samba_gplv3 (last visited Apr. 4, 2008). Samba is a program that allows Linux computers to interact via the network with Windows machines. *See* Samba, What is Samba?, http://us.samba.org/samba/what_is_samba.html (last visited Apr. 4, 2008).

203. SugarCRM, SugarCRM Announces Adoption of GPL v3, http://www.sugarcrm.com/crm/about/press-releases/20070725-GPLv3-FOSS.html (last visited Apr. 4, 2008). SugarCRM is a company that develops open-source based Customer Relationship Management (CRM) software. *See* SugarCRM, About Us, http://www.sugarcrm.com/crm/about/about-sugarcrm.html (last visited Apr. 4, 2008).

204. Shankland, *supra* note 131.

205. Tom Sanders, *Sun Tiptoes Into GPLv3*, VNUNET.COM, Nov. 15, 2007, http://www.vnunet.com/vnunet/news/2203486/sun-tiptoes-gplv3 (last visited Apr. 4, 2008).

cause of the uncertain legal ramifications of adopting GPLv3, and for fear of losing important patent portfolios. However, the patent provisions both hurt and help potential licensors of GPLv3, by making an express patent license while at the same time granting protection from patent retaliation by the licensee. Therefore, companies will likely think strategically and comprehensively about the result of adopting GPLv3 and whether it fits with their business goals. Ultimately, until GPLv3 becomes industry custom, as GPLv2 has already become, there will be commercial opposition to GPLv3.

VI. CONCLUSION

The most popular FOSS license in existence today, GPLv2 was released nearly sixteen years ago. During that period, significant legal and technological changes have emerged, exposing a significant amount of uncertainty and risk in the license. While GPLv3 clarifies existing ambiguities in GPLv2, including the scope of the copyleft obligation and dynamic linking, it also introduces new ambiguities. Businesses involved in FOSS looking for a much clearer delineation of their legal responsibilities compared to GPLv2 will be disappointed with GPLv3. Further, the sweeping patent provisions in GPLv3 are extensive and complex. Whether they are distributors of open source or proprietary software, technology companies should carefully examine the ramifications of these provisions for their patent portfolios and potential patent infringement actions. Lastly, according to Richard Stallman, GPLv3 also addresses perceived threats to free software such as Tivoization and DRM, although compromises were made to satisfy corporate interests. GPLv3 is an improvement on GPLv2, but it faces a tough road ahead to unseat GPLv2 as the premier FOSS license.