Journal of Competition Law & Economics, 8(3), 469–506
doi:10.1093/joclec/nhs013
Advance Access publication 10 July 2012

VIRAL OPEN SOURCE: COMPETITION VS. SYNERGY

Michal S. Gal[*]

ABSTRACT

The creation of free and open source software (FOSS) through social networks has been celebrated as one of the most interesting and inspiring developments of the information age. The main legal platform selected for facilitating this collaborative creation is the GNU General Public License (GPL). Software released under the GPL enables anyone to use, modify, and distribute the code. Yet, these rights are contingent upon virality: every copy or work based on the original code must also be subject to such terms and conditions. This article analyzes the interesting and intricate effects of virality on welfare and innovation. Virality increases motivations for parallel innovation, both in open source and in commercial code, *inter alia* by facilitating competition among networks and by preventing commercial firms from appropriating FOSS. At the same time, by almost closing the door on synergies between FOSS and commercial technologies, it limits cumulative innovation based on synergy and interoperability. As shown, FOSS creates an even stronger anti-commons tragedy than the patent regime. Virality's (non)regulation will thus determine the balance, as well as the connecting bridges, adopted by society between the two modes of production as well as between competition and synergy. While this issue arises in other contexts, the unique features of the software industry and of FOSS raise complex challenges. This article then analyzes market and legal responses to the GPL's virality. Such analysis is timely given that the viral GPL has become standard in many socially produced FOSS projects.

JEL: K21; K29; L21; L23; L24; L44; O31

I. INTRODUCTION

The creation of free[1] and open source software (FOSS) through social networks has been celebrated as one of the most interesting and inspiring

[*] Professor, Vice-Dean, and Director of the Forum for Law and Markets, University of Haifa Faculty of Law; Visiting Global Hauser Professor, New York University School of Law. Email: mgalresearch@gmail.com. Many thanks to Adi Ayal, Orna Agmon Ben-Yehuda, Yoav Alkalay, Rachel Aridor, Dan Crane, Dalit Ken-Dror, Andy Gavil, Niva Elkin-Koren, Haim Ravia, Michal Tsur, and participants at the Haifa/Loyola conference for excellent comments and/or fascinating discussions; to Oshrit Aviv and Avital Bruker for excellent research assistance. All views are explicitly my own.

[1] The term "free" is used in the context of open source to connote the freedom to use and modify the software, rather than its price: "free as in free speech, not free beer." RICHARD

phenomena of the information age.[2] Indeed, the collaboration of numerous software developers, contributing voluntarily and without immediate monetary benefit to the development of better software to be used and modified freely by all, is an inspirational story. It is a story that challenges some of our notions about what motivates people to share resources such as time, skill, and effort and our notions about how markets work. It is a story that is told, albeit with different emphasis, by libertarians and anarchists, anti-market activists and free market advocates alike.[3]

FOSS projects have already managed to challenge some deep-rooted market structures and behaviors. Linux, the world's largest FOSS platform, engages over 5,000 developers and is used by at least 2.4 million users.[4] It is estimated that 5.1 percent of the worldwide market for operating systems use Linux.[5] MySQL has approximately 20 percent market share in the market for database installations worldwide.[6] Some markets are even dominated by FOSS. Apache holds 66.9 percent of the world market for web servers,[7] and Microsoft has shrunk its investments in commercial alternatives. Indeed, as Raymond observed, the "bazaars" have shaken some of the

M. STALLMAN, FREE SOFTWARE, FREE SOCIETY: SELECTED ESSAYS OF RICHARD M. STALLMAN (Joshua Gay ed., Free Software Foundation 2002).

[2] For analysis of open source see, e.g., Daniel B. Ravicher, *Facilitating Collaborative Software Development: The Enforceability of Mass-Market Public Software Licenses*, 5 VA. J.L. & TECH. 11 (2000); ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR: MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY (2000), *available at* http://www.catb.org/~esr/writings/cathedral-bazaar/cathedralbazaar; David McGowan, *Legal Implications of Open-Source Software*, 2001 U. ILL. L. REV. 241 (2001); Yochai Benkler, *Coase's Penguin, or Linux and the Nature of the Firm*, 112 YALE L.J. 369 (2002); Greg R. Vetter, *The Collaborative Integrity of Open-Source Software*, 2004 UTAH L. REV. 563 (2004); Greg R. Vetter, *"Infectious" Open Source Software: Spreading Incentives or Promoting Resistance?*, 36 RUTGERS L.J. 53 (2004); Jonathan Zittrain, *Normative Principles for Evaluating Free and Proprietary Software*, 71 U. CHI. L. REV. 265 (2004); Brian W. Carver, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443 (2005); Ronald J. Mann, *Commercializing Open Source Software: Do Property Rights Still Matter?*, 20 HARV. J.L. & TECH. 1 (2006); CHRISTOPHER M. KELTY, TWO BITS: THE CULTURAL SIGNIFICANCE OF FREE SOFTWARE (Duke University Press 2008); Jonathan M. Barnett, *The Host's Dilemma: Strategic Forfeiture In Platform Markets For Informational Goods* 124 HARVARD L. REV. 1861 (2011).

[3] Niva Elkin-Koren, *What Contracts Can't Do: The Limits of Private Ordering in Facilitating a Creative Commons*, 74 FORDHAM L. REV 375 (2005).

[4] As of April 2010. THE FORMAL LINUX COUNTER REPORT, *available at* http://linux.derkeiler.com/Newsgroups/comp.os.linux.misc/2010-05/msg00001.html (last visited Dec. 10, 2011). Other sites have different estimates. *See, e.g.*, THE NUMBER OF LINUX USERS, *available at* http://www.numberof.net/number-of-linux-users/ (last visited Dec. 10, 2011).

[5] OS STATISTICS (Dec. 10, 2011), *available at* http://www.w3schools.com/browsers/browsers_os.asp.

[6] Trefis Team, *Oracle Will Use MySQL to Take Share from Microsoft*, TREFIS, *available at* https://www.trefis.com/company?article=12891# (last visited Dec. 10, 2011).

[7] W3Techs, Usage Statistics and Market Share of Apache for Websites, http://w3techs.com/technologies/details/ws-apache/all/all (last visited Dec. 10, 2011).

"Cathedrals."[8] FOSS is encouraged and used by governments in many countries, including Germany, the United States, France, and China. Leading commercial technology firms such as IBM, Google, Amazon, Sun, Java, and Oracle, and recently even Microsoft have also embraced it.[9] The open source movement has also inspired other industries to adopt somewhat similar models, one major example being Creative Commons, which applies to copyrighted materials. It is thus safe to say that FOSS is here to stay, and is likely to continue to shape social and market interactions.

FOSS is not just a social or technological phenomenon—it is also a legal one. FOSS builds on the existing proprietary system of copyright and contract law in order to create a set of legal rights and obligations that are attached to the source code. While licenses for the use of FOSS vary, most FOSS projects are licensed under the GNU General Public License (GPL), or relatively similar versions. The GPL creates a unique bundle of rights, authorizing anyone to copy, modify, and distribute the FOSS as long as any copies or derivatives based on the original code are distributed subject to the same license—a condition that has come to be known as virality.[10]

As this article elaborates, virality creates an interesting and intricate tradeoff in the way it affects competition and dynamic efficiency. On the one hand, virality increases incentives of developers to contribute to FOSS, *inter alia*, by preventing the appropriation of the code and by facilitating its dissemination. It also increases competition between commercial and viral FOSS codes as well as between networks based on such codes, thereby increasing motivations for parallel and competing innovation. On the other hand, virality creates barriers to innovations based on synergies between different technologies in the computer ecosystem, given that it significantly limits the ways that investments in synergies can be commercially appropriated. Such effects might be especially significant where a market's infrastructure is largely based on either viral FOSS or commercial software, and where innovative and significantly welfare-enhancing ideas embedded in viral FOSS or in commercial software cannot be easily duplicated. Furthermore, virality even limits some synergies between FOSS projects. The welfare effects of lost synergies might be significant as software innovations often build upon previous ones or require interconnections with them.

[8] RAYMOND, *supra* note 2.

[9] *See, e.g.*, Stephen M. Maurer, *The Penguin and the Cartel: Rethinking Antitrust and Innovation Policy for the Age of Commercial Open Source*, UTAH L. REV. (forthcoming Spring 2012); Greg R. Vetter, *Commercial Free and Open Source Software: Knowledge Production, Hybrid Appropriability, and Patents*, 77 FORDHAM L. REV. 2087 (2009); Barnett, *supra* note 2.

[10] The term was first coined by Margaret Jane Radin and relates to contracts which limit the rights of subsequent owners or users of the property. Margaret Jane Radin, *Humans, Computers and Binding Commitments*, 75 IND. L.J. 1125 (2000). When applied to knowledge, the effects of such virality may be different than when applied to physical chattels. This is because it may apply to a much larger group given the public good characteristic of knowledge.

Indeed, following Maurer and Scotchmer, viral FOSS might be harmful to society if it displaces other tools that are needed to extract the full value from society's investment in knowledge.[11]

Notably, the dilemmas surrounding access to or exclusion from past innovations are not new. Rather, they are a central and persistent feature of questions regarding the optimal scope of intellectual property rights and the scope of limitations to be imposed on monopolists refusing to grant access to their products or services. Both cases, as well as viral FOSS, raise the need to balance the incentives for *ex ante* investments in research and development with the *ex post* ability to innovate by building upon previous innovations. Yet, due to its unique nature, viral FOSS creates the most complex effects on competition and innovation.

Interestingly, virality epitomizes—and strengthens—the clash between traditional models of production, which are based on personal economic profit, and new social production models, which are based on wider incentives, including self-satisfaction, ideology, social recognition, and a wish to take down the "cathedrals" created by the traditional production model. Virality's (non) regulation will thus determine the balance, as well as the connecting bridges, adopted by society between the two production models in software markets. Accordingly, this article attempts to untangle the effects of virality on social welfare, in order to determine the optimal legal regime that should apply to it.

The rest of the article is structured as follows. Part II sets the stage by reviewing the *raison d'être* as well as the *modus operandi* of the FOSS movement, with a special focus on viral licenses. Part III examines the welfare effects of viral licenses. The findings shake some of the assumptions often made about the welfare effects of viral FOSS and serve as a basis for an operational suggestion that better aligns licensing terms with social welfare considerations.

The widespread adoption of viral licenses has led market players to try and devise ways to overcome the obstacles created by virality on possible synergies. Part IV analyzes these strategies and evaluates their ability to achieve this goal, as well as their effects on social welfare. Given limitations on self-help strategies, this article analyzes regulation by antitrust and copyright in order to examine whether a legal framework can be used to reduce barriers to innovation that significantly reduce social welfare. The time is ripe to evaluate such responses, as viral licenses have become standard in many FOSS projects. As elaborated below, despite the spread of new types of licenses, a large percentage is still viral.

[11] Stephen M. Maurer & Suzanne Scotchmer, *Open Source Software: The New Intellectual Property Paradigm* 28 (Nat'l Bureau of Econ. Research, Working Paper No. 12148, 2006), *available at* http://www.nber.org/papers/w12148.pdf.

II. THE RISE OF FOSS AND ITS LEGAL FRAMEWORK

A. The FOSS Movement

The FOSS movement is a multifaceted phenomenon. It is a distinct way of writing and sharing software, giving way to new socio-economic structures in the way innovation is spurred and production is organized. It also offers an innovative legal strategy for achieving economic and ideological goals.

FOSS began as a rebellion against proprietary software where a company keeps the source code of its applications secret (or "closed"), thereby making it difficult for others to change it, even for their own non-commercial purposes.[12] As Elkin-Koren notes, the open source movement is nothing short of a social movement that aims at changing social practices and social norms related to software. It advocates the use of legal rights in a way that changes their meaning. Developers are called to voluntarily restrain the power they were granted under copyright law for the benefit of society at large. Ultimately, the purpose is to redefine social norms and promote the values of sharing and reusing. The vehicle for such social change makes use of existing legal frameworks of copyright and contract to create a new form of copyright licenses.[13]

This social movement also brought about a new mode of software production, based on collaborative and voluntary social production, which was no longer confined to corporate and institutional borders. The first project to employ such a method was a FOSS operating system, Linux. Linux was initiated by Linus Torvalds, but was further developed by a virtual community of software developers from around the world. This community was connected mainly through the internet and donated time and effort outside their employment agreements.[14] Following this model, many other FOSS projects were based on volunteerism-centric development, in whole or in part. The motivations of contributors to FOSS are diverse, and include non-market motivations, such as social interactions via cooperative creative activity, being part of a gift economy, and the thrill of taking on proprietary software giants, as well as market motivations, including reputation and experience.[15] In addition, many developers are motivated by the philosophy of software freedom, according to which source code should be available for copying, modification, and subsequent distribution in order to facilitate, *inter alia*, software innovation and improvements.

[12] For further information about the Open Source Movement visit, *inter alia*, The Open Source Website, http://www.opensource.org/ (last visited Dec. 5, 2011). *See also* STALLMAN, *supra* note 1.

[13] Niva Elkin-Koren, *Exploring Creative Commons: A Skeptical View of a Worthy Pursuit*, *in* THE FUTURE OF THE PUBLIC DOMAIN 325 (P. Bernt Hugenholtz & Lucie Guibault eds., Kluwer Law Int'l 2006).

[14] Niva Elkin-Koren, *Tailoring Copyright to Social Production*, 12 THEORETICAL INQUIRIES L. 309, 318–22 (2011).

[15] Such motivations are elaborated in Part II.D below.

The transparency of the code, the modularity of its development, and the low costs of online communication infrastructure enabled cooperation on a scale not envisioned before, which was not limited by the traditional boundaries of corporate entities or by significant informational obstacles. This was a revolution in software production. As elaborated below, this production model enabled developers to identify bugs and fix them promptly as well as add new functions to the software, sharing the information with the entire community and making corrections and additions immediately available to the public, without engaging in costly transactions.[16] This, in turn, created lower cost software with a serious shot at better quality, which then attracted more users and developers.[17] As Benkler argues, collaborative FOSS is the "quintessential instance of commons-based peer production."[18]

Today, commercial companies fully or partially fund, support, and govern many social-production FOSS projects.[19] This symbiosis, which is part of the growing phenomenon of a hybrid economy, can be beneficial for both sides. FOSS benefits from funding of activities that would not otherwise be supplied (for example, costly on-going monitoring of code modifications or developments of code modules that could not be developed by way of modularity), by the ability to use proprietary code donated by commercial firms, and from the funding of additional developers to develop FOSS.[20] Diverse motivations cause commercial firms to support FOSS. These may include needling your rivals by supporting a competing, free software,[21] creating a reputation for fairness and generosity, and gaining a customer base and knowledge of potential uses of one's commercial software.[22] As Barnett suggests, supporting FOSS may also be used to overcome host commitment problems in platform markets. Creating a commoditized FOSS platform reduces the fears of potential users of exploitation of the platform by its owner, and enables its creation and growth. The supporting firm can then profit from complementary goods and services: hardware, software, and services.[23] Another type of strategic conduct, which embraces FOSS for

[16] YOCHAI BENKLER, THE WEALTH OF NETWORKS: HOW SOCIAL PRODUCTION TRANSFORMS MARKETS AND FREEDOM (Yale Univ. Press 2006).

[17] Carver, *supra* note 2.

[18] BENKLER, *supra* note 16. For information on the open source movement see, e.g., The Open Source Website, *supra* note 12.

[19] *See, e.g.*, Vetter, *supra* note 9; Barnett, *supra* note 2.

[20] *See, e.g.*, Barnett, *supra* note 2, at 1893.

[21] IBM, for example, announced that it would not enforce some of its patent portfolio against Linux users, thereby strengthening the operating system that seemed to have the best potential to compete effectively with Microsoft's Windows operating system. *See, e.g.*, DON TAPSCOTT & ANTHONY D. WILLIAMS, WIKINOMICS: HOW MASS COLLABORATION CHANGES EVERYTHING 30 (Penguin 2007).

[22] Courts recognized the economic motives in public licenses, even where the profit is not immediate. *See, e.g.*, Jacobsen v. Katzer, 535 F.3d 1373 (Fed. Cir. 2008).

[23] Barnett, *supra* note 2.

one's own purposes, involves active participation in FOSS development to ensure that it makes the best use of one's comparative advantage.[24]

Notably, the licensing platforms used for such commercially supported FOSS projects generally do not include viral terms. Still, commercial firms also support some viral FOSS projects, such as Linux. This may occur, for example, where the sole purpose of the support is to needle one's rivals, or where virality does not limit interoperability with one's compatible products (for example, because the commercial firm sells services).[25] Yet, as elaborated below, virality might often limit the supporter.

FOSS has thus outgrown its humble origins. As it continues to develop, so does the motivation to use it or, at least, to create compatibility with it. The hybrid economy, which involves both FOSS and commercial software firms, is here to stay.

B. The Legal Framework: GPL

Key to FOSS' immense success was its legal design.[26] While FOSS projects employ a wide range of licenses, the GPL is by far the most commonly used, and thus is the focus of this article. The GPL, the first version of which was written in 1989 by the initiator of the open source movement, Richard Stallman, has become the standard license for most FOSS projects; despite its slow decline in recent years, it is still used by more than 56 percent of the market for FOSS.[27] Yet this percentage does not indicate the GPL's real effect, which stems from the importance of several large projects that adopted it, such as Linux and MySQL, as well as from the fact that the GPL often acts as a benchmark, and thus its major obligations have spill-over effects into other licenses. Indeed, according to Benkler, more than 85

[24] IBM provides an interesting example. IBM organized a FOSS project, which involved a special team of its developers as well as peer volunteers, designed to create FOSS software that makes best use of IBM computers' unique capabilities of auto-vectorization in order to increase its comparative advantage in the market for computer hardware. This strategy disconnects the purchase of software from hardware. Although consumers might pay for IBM's investment through the price of the hardware, perception plays an important role, since one of the achievements of the FOSS revolution is a change of perception of many users—why buy software if you can receive a competing one for free? Moreover, buyers acknowledge that IBM would have a strong incentive to continue and develop the FOSS that is compatible with and maximizes the use of its hardware. The project also allows IBM to harness the power of the crowd to its own benefit.

[25] Interestingly, Koski found that the more the firm is service oriented, the more it will be likely to offer products using open source (but not necessarily viral) licenses. Heli Koski, *OSS Production and Licensing Strategies of Software Firms*, 2 REV. OF ECON. RES. ON COPYRIGHT ISSUES 111 (2005).

[26] For the GPL's creation and history, see, e.g., John Tsai, *For Better or Worse: Introducing the GNU General Public License Version 3*, 23 BERKELEY TECH. L.J. 547, 548-53 (2008).

[27] Open Source Resource Center, http://osrc.blackducksoftware.com/data/licenses/index.php (last visited Dec. 5, 2011).

percent of active FOSS projects include some version of the GPL or a similarly structured license.[28] These facts, in themselves, indicate an important potential benefit of the GPL: it creates standardization and thus reduces transaction costs that would arise from studying the terms of each license.[29] Standardization is strengthened by the fact that the GPL is supported by the Free Software Foundation (FSF), which provides legal support on an on-going basis, thereby increasing the unified interpretation of the GPL.[30] Yet, as elaborated below, this very trait might also increase its negative welfare effects if the market is locked-in into a sub-optimal license or a sub-optimal interpretation of the license.

The GPL, which is a non-exclusive license, has adopted many concepts of public domain licenses, with the added twist of virality.[31] It has been updated twice, but its basic principles still stand.[32] We shall refer to version 2, which is used in many major FOSS projects.

The legal strategy of the GPL for promoting the sharing and reuse of software rests on three pillars: copyright in the code, free and open source code, and virality. The GPL asserts copyright in the code, thereby enabling the licensors to determine the terms of use and to prevent others from capturing the code and making it proprietary.[33] The other two components establish the unique "deal" that the GPL offers to potential users. This "deal" offers significant benefits: software released under the GPL allows anyone to use, copy, modify, and distribute the code. The granting of such a wide license, which encourages imitation and use, is a major component of the FOSS ideology, as the GPL's preamble states: "The licenses for most software . . . are designed to take away your freedom to share and change the works. By contrast, the [GPL] is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users."[34]

[28] BENKLER, *supra* note 16, at 90.

[29] Benjamin Mako Hill, *Towards a Standard of Freedom: Creative Commons and the Free Software Movement*, ADVOGATO (July 29, 2005), *available at* http://www.advogato.org/article/851.html; Elkin-Koren, *supra* note 13.

[30] Such support raises interesting questions, such as the weight to be given to the FSF's interpretation of the GPL, especially when it applies to code in which the licensor does not have copyrights.

[31] Douglas A. Hass, *The Myth of Copyleft Protection: Reconciling the GPL and Linux With the Copyright*, 25 A.B.A. INTELLECTUAL PROP. L. NEWSL. (2006).

[32] The first version, written by Richard Stallman, a computer scientist, was published in 1989. The second version, to which the legal scholar Eben Moglen contributed, was published in 1991. Some downsides of the GPLv2 generated discussions in the FOSS community which led to the adoption of GPLv3 in 2006.

[33] David McGowan, *Legal Aspects of Free and Open Source Software*, *in* PERSPECTIVES ON FREE AND OPEN SOURCE SOFTWARE 211, 214-7 (Joseph Feller, Brian Fitzgerald, Scott A. Hissam & Karim R. Lakhani eds., MIT Press 2005).

[34] GNU GENERAL PUBLIC LICENSE, VERSION 2, Preamble (Free Software Foundation, June 1991).

To facilitate such modifications, the GPL mandates the distribution not only of the object code—which consists of binary instructions to a computer on how to operate the software—but also of the source code: such instructions in a computer programming language (e.g., C++, Java, BHP). The benefits are clear: open source makes it much easier for software developers to understand the code, and use their freedom to modify it or write applications to it. Such modifications range from minor bug fixes to the reuse of source code in entirely different projects.[35]

In exchange for such rights, the user is subject to virality: "You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License."[36]

Accordingly, any distributor of the code, a derivative work, or any work "based on the code" must release his *entire* work under the GPL[37] and be subject to the same contractual terms as the original work. The GPL's preamble justifies this limitation as follows: "To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights."[38] The GPL includes no exceptions to this condition. On first blush, virality seems like a simple reciprocal commitment: if you use my free and open code, you should also make your code, which makes use of mine, free and open to everyone. Yet, as elaborated below, this "grant-forward" condition, which applies beyond the relationship between the parties, does not always ensure an exchange of things of equal value.

It is noteworthy that the GPL does not limit the use of FOSS for internal needs in private computer ecosystems and in development processes, so long as the FOSS code does not constitute an integral part of the commercial product. For example, a commercial firm can use viral FOSS for quality testing of its products or for analyzing the results of such tests. This results from the fact that virality's triggering event is not using or modifying of the code, but rather its redistribution.

The GPL also does not place any limitations on the price that can be charged for the code. Rather, its freedom relates to the freedom to use, redistribute, and modify the code. Yet because everyone is free to redistribute

[35] Heidi S. Bond, *What's So Great About Nothing? The GNU General Public License and the Zero-Price-Fixing Problem*, 104 MICH. L. REV. 547 (2005).

[36] GNU GENERAL PUBLIC LICENSE, VERSION 2, § 2(b). Section 6 of the GPL provides that each time a GPL licensee redistributes "the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program."

[37] An important question is whether a copyright owner is allowed to tell others what they can do with their own copyrighted material, since at least part of the new code might belong to the creator of a derivative work. This question is beyond the scope of this paper. We assume that it is allowed in some circumstances.

[38] GNU GENERAL PUBLIC LICENSE, VERSION 2, Preamble.

the code, its price will tend to be the market price for distribution; anyone selling the code for a higher price can be easily outbid. [39]

III. THE WELFARE EFFECTS OF VIRALITY

As viral FOSS projects become more widespread, the need to critically evaluate their welfare effects increases to ensure that their use of intellectual property rights indeed serves the goals that these laws were set to achieve. The following discussion analyzes both the positive and negative welfare effects of virality. As we shall see, virality creates a unique tradeoff between cumulative and parallel innovation, thereby challenging some of our assumptions about the most efficient way to increase dynamic efficiency.

The analysis proceeds in two stages. First, potential welfare effects of FOSS are identified to make the case that virality might be facilitating welfare-enhancing software. The next step is to analyze the effects of virality on the creation of welfare-enhancing FOSS.

For the purposes of this part, we assume a strong and wide-scoped virality condition, in line with the way the GPL is interpreted by the FSF. This view will be challenged in the next part. In addition, the analysis focuses on economic welfare considerations only and gives little weight to other values such as moral rights in copyright.

A. Potential Welfare Effects of FOSS

FOSS has many potential positive social welfare effects. As this has been the focus of other studies, the analysis below is brief. Because of the code's openness and freedom of use, FOSS reduces costs of solving problems in computer ecosystems.[40] It facilitates the creation of a public repository of solutions to common problems, thereby reducing the need to replicate existing code and reinvent the wheel. Its open and free nature also increases interoperability.[41]

FOSS also creates significant benefits in education. Its openness facilitates the study of coding and software technology at every level of complexity and in a diverse array of information technology environments.[42] FOSS is also subject to fewer viruses, given the ideology of most hackers and their potential social response to attacks on FOSS. In addition, the openness of

[39] Bond, *supra* note 35.

[40] *Id.* at 110; Michal Tsur & Shay David, *A License to Kill (Innovation)? Open Source Licenses and Their Implications for Innovation* 27 (Apr. 30, 2005), *available at* http://ssrn.com/abstract=858104.

[41] Fershtman and Gandal find empirical support for the existence of knowledge spillovers among open source projects. Chaim Fershtman & Neil Gandal, *Direct and Indirect Knowledge Spillovers: The "Social Network" of Open Source Software,* 42 RAND J. ECON. 70 (2011).

[42] Bond, *supra* note 35; Tsur & David, *supra* note 40.

the code dramatically reduces consumer switching costs if the code-provider fails mid-way through a project.[43]

FOSS projects also potentially spur innovation and increase software quality through sharing and collaboration. This is especially true where FOSS is created through social production. The ability to bond the creative minds of a large number of developers from different backgrounds to focus on a given project may create a level of development and innovation that no commercial firm, operating alone, could achieve. As Raymond has put it, "with enough eyeballs all bugs are shallow."[44] Furthermore, relative to an organized production model, where assignments are imposed from higher levels of management, a social production model, where each developer contributes where, when, and what he wishes, may be more efficient.[45] This is because each developer is best aware of his comparative advantages as well as his passions and preferences. Quality may also increase due to the rapid pace of FOSS development. This advantage becomes apparent when considering that, for commercial firms, the payoffs from making clever but rapid improvements to existing software may be low relative to the costs of new releases. Quality also benefits from the fact that, to a large extent, social production FOSS are user-generated platforms. The developer is often also a user; therefore, the informational obstacle about what users want is significantly reduced. While decisions where to contribute are made by each developer privately, trends in contributions signal joint demand patterns while at the same time creating supply. This may lead to the incorporation of new features and to the provision of better solutions to existing problems.[46]

It is noteworthy, however, that FOSS does not always increase quality. Empirical evidence on code quality is mixed, although many high-profile FOSS projects are clearly of high quality. This might result, *inter alia*, from motivational limitations on the goodwill of developers to donate innovative ideas to FOSS, the limited ability to develop a specific code via modularity, or the suboptimal incorporation of ideas in FOSS. Yet as long as the software does not enjoy first-mover advantages and switching costs are not prohibitively high, we can rely on the market mechanism of consumer demand to reflect the value of FOSS to users, relative to its alternatives.[47]

Accordingly, FOSS can potentially create positive welfare benefits depending, *inter alia*, on the market conditions. Let us now analyze the

[43] Hal R. Varian & Carl Shapiro, *Linux Adoption in the Public Sector: An Economic Analysis* (Working Paper 2003), *available at* http://people.ischool.berkeley.edu/~hal/Papers/2004/linux-adoption-in-the-public-sector.pdf.

[44] RAYMOND, *supra* note 2.

[45] BENKLER, *supra* note 16; Maurer & Scotchmer, *supra* note 11, at 22.

[46] Maurer, *supra* note 9.

[47] *See also* CLAY SHIRKY, HERE COMES EVERYBODY: THE POWER OF ORGANIZING WITHOUT ORGANIZATIONS 246-47 (2008).

effects of virality on creating welfare-increasing FOSS: is virality the best way to achieve such benefits?

B. Possible Positive Effects of Virality

Two potential major positive effects of virality can be identified. They are strengthening the incentives of developers to contribute to FOSS projects and strengthening incentives of non-viral FOSS firms to compete.

Let us start with incentives to contribute to FOSS. The importance of such incentives should be downplayed, since maintaining the enthusiasm and the sense of trust among potential volunteer developers is an important component of the success of collaborative FOSS projects.[48] Virality strengthens such incentives in two ways. First, it facilitates the assimilation of FOSS. Even those who would otherwise not have adopted virality for their own code might do so if they wish to use viral FOSS in their own products.[49] Furthermore, the combination of virality and the nature of software creates a feedback effect. Software often benefits from interoperability, rendering a piece of software more useful when it is compatible with other technologies. As the number and quality of viral FOSS increases, so does the incentive to use or interoperate with it. Thereby, FOSS grows much faster than without virality. As Carver observed, the self-perpetuating feature of the GPL is likely primarily responsible for its widespread adoption.[50] This, in turn, strengthens the motivation of developers to take part in FOSS creation: it boosts the motivation of those who aim to create a world in which all source code is free and open; it strengthens those motivated by their own use of the FOSS, by increasing its value to them; and it motivates purely innovation-related developers as it creates a growing platform to which they can contribute.

Second, virality forecloses the proprietary appropriation of FOSS.[51] It prevents users from benefitting from the incorporation of FOSS in their products without compensating developers. As Maurer explains,

> [a] most profitable business strategy is to copy the underlying [FOSS] module at zero cost and then sell improvements for whatever the market will bear. This, of course, is a formula for disaster: If every company does this [FOSS] will disappear entirely. Viral provisions block this outcome by forcing companies that use [FOSS] code to donate their improvements back to the community.[52]

By preventing unjust enrichment and misappropriation by free riding, virality strengthens motivations to contribute to its creation. Such a stimulant

[48] Elkin-Koren, *supra* note 14.
[49] Carver, *supra* note 2, at 447.
[50] *Id.* For a similar conclusion, see Tsai, *supra* note 26.
[51] Vetter, *supra* note 9.
[52] Maurer, *supra* note 9.

may be especially important where it is based on the wish to break down dominant commercial firms.

Moreover, virality operates as a commitment device by ensuring that FOSS would continue to develop as a free and open source code as long as developers are willing to contribute to it, thereby increasing motivations of users to choose it over proprietary software.

Another potential positive welfare effect of virality is the procompetitive pressure it creates on commercial software suppliers. Most importantly, it might create competition in markets where scale economies or network effects are significant, because some of FOSS' characteristics make it easier to reach significant scales. Foremost, where FOSS is created, fully or partially, by social production, the internalized production costs are low, and FOSS enjoys an important advantage over commercial software that must bear all costs of development. Second, its free and open nature makes it attractive to many. Third, its availability over the internet and the ease of licensing reduce transaction costs. Indeed, in the last decade, FOSS projects have put tremendous procompetitive pressure on their proprietary rivals. There is a FOSS alternative, and usually a pretty good one, to just about every major commercial software product. For example, Linux challenges Microsoft's Windows (mainly in the servers' market) and Mozilla Firefox web browser was the first to pose a formidable challenge to Microsoft's Internet Explorer. This, in turn, forces commercial firms to find better, competing solutions. Another way in which FOSS may enhance dynamic efficiency is by reducing entry costs for firms that supply complementary products and services where a software market was previously dominated by a commercial firm, thereby increasing the likelihood that such services will be competitively supplied.[53]

Such competitive pressures are strengthened by the unique model of market interaction advanced by virality. This creates (at least) two separate, competing spheres: viral FOSS and commercial software. These two spheres interact mainly by competing over the consumer in product and services markets. By closing the door on synergies, as elaborated below, virality strengthens motivations of commercial firms to develop other solutions to similar problems, thereby potentially increasing parallel innovation.

C. Possible Negative Welfare Effects of Virality

Yet virality can potentially limit innovation and harm social welfare in at least one important way: it limits synergies that could have increased social welfare.[54] While it generally does not prevent user-created synergies, it might prevent synergies created by the developer for use by others.

[53] R.E Hawkins, *The Economics of Open Source Software for a Competitive Firm: Why Give it Away for Free?*, 6 NETNOMICS 103 (2004).

[54] Tsur & David, *supra* note 40, at 32; Vetter, *supra* note 2.

Synergies are created when the value of a product which incorporates different inputs is higher than the value of each input standing alone. In economic terms, assume that Product *A* increases social welfare by $50 per unit, as does Product *B*, whereas a combined product of *A* and *B* increases dynamic efficiency significantly, thereby increasing social welfare by $500. The combined product can be a chattel (for example, a cell phone), a service (for example, tech support), or software. Synergies can potentially increase efficiency significantly, either by creating higher value products or by reducing production costs. Synergies are technologically relatively easy to create among software, given that software is a public good that can be easily replicated at low costs. As information technology progresses, more and more products are (or can potentially be) compound items, made of many components incorporating different pieces of technology, often created by different producers and based on past innovations.[55] Thus, the ability to borrow and share past innovations significantly increases dynamic efficiency.

Two types of synergies can be identified. The first involves a synergy of separate yet complementary products that have a joint interface based on *interoperability*, such as an operating system and the software that can run on it. The second involves the creation of a *whole new product*, combining both inputs ("second generation innovation"). As elaborated in the next part, while the market has found some ways to overcome obstacles to synergy of FOSS and commercial products by way of interoperability, the market has rarely found solutions for synergy by way of a compound product.

For synergies to take place, two conditions must be met: first, the relevant parties must be aware of possible synergies ("the informational obstacle"). Second, all relevant parties must find it worthwhile to invest in the realization of potential synergies ("the motivation obstacle").

FOSS reduces the informational obstacle, regardless of whether it is viral or not. Its openness allows anyone to observe the code, as long as he meets certain conditions stated in the license. The fact that most FOSS is available through the internet lowers the informational obstacle even further. In contrast, the source code of proprietary products often cannot be observed. Indeed, it was the "closed" code of the driver of a new printer—which did not enable its users to alter it in a way that would increase the benefit derived from it—that motivated Stallman to start the first FOSS venture. Accordingly, developers seeking synergies may find innovations developed in FOSS rather easily.

At the same time, relative to FOSS developers, proprietary developers might have stronger incentives to actively seek possible synergies in order to maximize their investment. This results from the fact that the search costs of

[55] Maurts Dolmans & Carlo Piana, *A Tale of Two Tragedies: A Plea for Open Standards*, 2 OPEN SOURCE SOFTWARE L.J. 115 (2010).

each FOSS developer may well be higher than his personal gain from locating possible synergies. Still, much depends on the model in which FOSS is created and applied. If, for example, the relevant FOSS community invests in seeking and publishing information on possible synergies, such a top-down organized effort may disperse search costs over numerous developers and reduce the informational obstacle. External funding can also overcome this obstacle. Furthermore, since not all synergies can be predicted by the software developer, the transparency of FOSS allows more people who are involved in innovation processes in the market to identify possible synergies.

While FOSS generally reduces the informational problem, it aggravates the motivational obstacle in some situations, and reduces it in others. Below we explore two paradigmatic cases. In the first case all synergetic products are FOSS. In the second case one product is viral FOSS and the other is commercial.

1. All Synergetic Products as FOSS

Let us first explore a situation in which all products are FOSS. The FOSS nature of the product significantly reduces motivational obstacles, since there is no need to invest in contracting and buying any of the synergetic components. Yet two obstacles to synergy might still emerge, one motivational and one legal. Where FOSS is based on a model of voluntary contribution by private developers, and the synergy requires a large investment that could not be spread over developers, the private benefits to any developer who invests in creating the synergy might be higher than his private costs. Furthermore, if, for example, most developers are motivated by own-use incentives, this might lead to under-investment in synergies since the developer does not directly benefit from the positive externalities created by the synergy. This implies that even if the social welfare effects of the synergetic product are high and are much larger than the overall costs in creating the synergy, it would not always be realized.[56] External funding to overcome the collective action problem, as well as internal leadership and guidance of developers to invest in such synergies, reduce such obstacles. A legal obstacle might arise when the licensing terms that cover the different FOSS clash, for example, where the licenses adopt different degrees of virality.

2. At Least One Product as Proprietary

Let us now explore a situation in which at least one of the products is viral FOSS and at least one other is commercial. The viral nature of a product significantly increases the motivational obstacle. This is because, for the synergy to accrue, it should be Pareto-optimal for all parties. The

[56] This is, of course, also true for commercial firms; however, in their case, the benefits might be higher and the collective action problem lower.

commercial firm will only invest if the expected profits from the investment are higher than the costs incurred. But virality significantly reduces its ability to profit from the synergetic product. Moreover, due to virality, its proprietary technology, which was embedded in the synergetic product, now also turns to viral FOSS and could only be distributed as such in any project (not just FOSS) in which it can potentially be used and distributed. The result resembles a reverse game of Topple: once you insert a certain block (viral FOSS), the whole structure (development for economic profit) topples. This may be a high price to pay, as indicated in the fact that commercial software firms often implement defensive strategies to ensure they do not incorporate viral FOSS in their products. Accordingly, for synergy to take place, the synergetic product would need to create value for the commercial owner of the synergetic code in other ways (for example, reputation or the provision of complementary services) that are often hard to create.[57] This, in turn, significantly reduces the incentives of commercial firms to invest in synergetic products and processes which involve viral FOSS.

A difference in the incentive structure between commercial firms and owners of viral FOSS further exacerbates this motivational problem. In the former case, the owner's decision whether to allow synergies is based on economic motivations—that is, which strategy would allow it to reap the largest reward. The potential market profit serves as a crude but important indication of the welfare benefits created by the new product or service. Accordingly, the commercial owner of the software will generally have incentives to invest in social-welfare-enhancing products or processes by either producing them himself or licensing others to do so. This is not true of viral FOSS, given that the welfare effects on the market do not necessarily translate into incentives to allow synergies. Thus, we cannot assume that socially-desirable synergies would generally be created.

Since we cannot assume that developers working in commercial firms always have better ideas than viral FOSS developers, the "next best idea" with a potential to create large synergies might be embedded in viral FOSS. Given that the nature of innovation in software markets is often complementary,[58] the effects on social welfare can be significant if potential synergies involve commercial software.

Indeed, the limitations created by virality on possible synergies produce what has come to be known a tragedy of the anticommons, which involves

[57] Indeed, Bonaccorsi and Rossi surveyed Italian firms that use open source software and found that, on average, firms that employ software with restrictive licenses supply fewer proprietary products than firms that employ software with less restrictive licenses. ANDREA BONACCORSI & CHRISTINA ROSSI, LICENSING SCHEMES IN THE PRODUCTION AND DISTRIBUTION OF OPEN SOURCE SOFTWARE: AN EMPIRICAL INVESTIGATION (2002), *available at* http://opensource.mit.edu/papers/bnaccorsirossilicense.pdf.

[58] Maurer & Scotchmer, *supra* note 11.

the under-use of private goods controlled by more than one right holder.[59] The argument was first developed in the context of a patent regime. The claim was that dispersed ownership of patents harms social welfare because they limit synergies. Yet viral FOSS creates an ever stronger anti-commons tragedy than a patent regime. This is because under a patent regime the originator of an idea can license its use to others and will have an incentive to do so if such licensing benefits him economically, and because patent protection is limited in time. In contrast, the extremity of the virality condition never enables such a synergy for commercial purposes.

Naturally, the scope of the effects of limited synergies cannot be directly observed, given that potential synergies might have been abandoned because of the FOSS' virality. Yet some indications of such possible synergies can be observed from those cases in which allegations arose that proprietary firms attempted to incorporate viral FOSS into their products. The recent allegations against Apple exemplify the possible effects of limited synergies, especially where dominant platforms are proprietary. Apple was accused of including a possibly viral software, GNU Go, in its application store. As a result, it immediately removed the software,[60] thereby preventing the ability of tens of thousands of existing viral software applications to be offered in Apple's application store for the iPhone.

Notably, while the GPL was not crafted with an objective to maximize overall software innovation,[61] the synergy problem was recognized by its creators, which also created a unique license, the so-called Lesser General Public License (LGPL). The LGPL is similar to the GPL, except that it includes an important exception: its virality does not apply to dynamic links between the code that it covers and a completely closed code. The exception was inserted in order to ensure that binary modules which are part of the computer's hardware, and which interact with the FOSS, would not be infected by the GPL's virality. This moderately restrictive alternative license was essential because many developers of such binary modules did not agree to release their code openly.

Virality can reduce competition and thus dynamic efficiency in another, related way, where the FOSS firm enjoys a monopoly position in its market. Suppose that firm *A* has a dominant position in software market A. Firm *B* wishes to compete in market B, which provides a complementary software to market A. Firm *B* faces two entry options: (1) contract with firm *A* for entry into market A; or (2) incur the high costs of entering market A as well. If

[59] *See, e.g.*, Michael A. Heller & Rebecca Eisenberg, *Can Patents Deter Innovation? The Anticommons in Biomedical Research*, 280 SCI. 698 (1998); MICHAEL HELLER, THE GRIDLOCK ECONOMY: HOW TOO MUCH OWNERSHIP WRECKS MARKETS, STOPS INNOVATION, AND COSTS LIVES (Basic Books 2008).

[60] Brett Smith, *GPL Enforcement in Apple's App Store,* FREE SOFTWARE FOUNDATION, May 25, 2010, http://www.fsf.org/news/2010-05-app-store-compliance (last visited Dec. 10, 2011).

[61] Tsur & David, *supra* note 40, at 22-23.

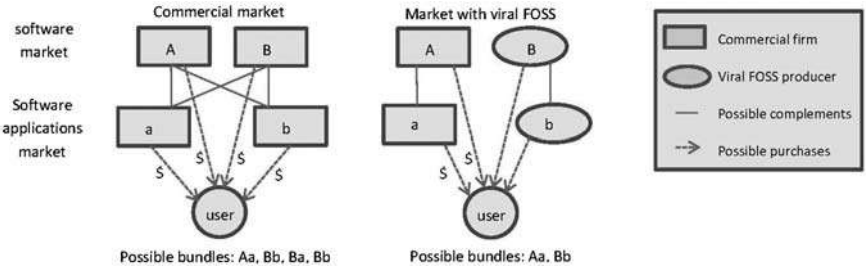

Figure 1. Possible consumer bundles under different market conditions

firm *A* is viral FOSS and firm *B* is commercial, the contracting option generally does not exist. Firm *B* must therefore enter both markets, if it wishes to compete. This implies that some firms will not enter market B, even if they can supply a more efficient product than is currently supplied in it. This entry obstacle might also lead to a situation in which market B will simply not exist.

To sum, viral terms produce incentives detrimental to synergy between FOSS and commercial code, even when the synergy can significantly contribute to the dynamic growth of the software industry and even to FOSS users.

Of course, the creation of obstacles to synergy and interoperability are not limited to viral FOSS. Microsoft provides an example of a commercial firm that attempted to protect its monopoly in one software market by erecting high obstacles to interoperability in interconnected software markets.[62] Yet commercial motivations might reduce the incentives of commercial firms to create obstacles to social-welfare-enhancing synergies. But, more importantly, the fact that commercial firms also attempt to use such strategies does not imply that viral FOSS should be automatically allowed to erect such barriers.

D. Partial Conclusion: The Tradeoff

Virality thus creates an apparent tradeoff. The two-separate-spheres-of-operation model created by it significantly limits the interoperability and synergy between the spheres, which could have potentially increased cumulative and even parallel but interoperable innovation. At the same time, the separate-spheres model creates inter-sphere competition, thereby strengthening incentives to innovate in both the viral FOSS sphere and in the commercial production sphere. It thus enlivens the debate on the best ways to achieve allocative, productive, and, most importantly, dynamic efficiency.

[62] *See, e.g.*, Case T-201/04, Microsoft Corp. v. Comm'n of the Eur. Cmtys., 2007 E.C.R. II-3601, *available at* eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri= CELEX:62004TJ0201:EN:HTML.

The challenge is thus to find the optimal balance between the clashing effects on social welfare analyzed above.

This tradeoff raises an important question of whether *strict* virality, which creates no exceptions, is indeed necessary in order to achieve the positive welfare effects of virality discussed above. If the answer is negative, then adopting a less strict condition can potentially increase social welfare. Indeed, as elaborated below, some of the positive effects of virality may not need strict virality, and strict virality might even inhibit these positive effects. Instead, a more moderate condition that enables the FOSS owner to create exceptions to virality where synergies create significant positive welfare effects may be justified.

Let us first carefully examine the effects of a more moderate virality condition on motivations to create FOSS. The first effect, identified above, focused on virality as a commitment device for ensuring that FOSS efforts are not appropriated. However, a more moderate virality condition can also achieve this. So long as the license includes a suitable compensatory model, of which developers are aware, this positive effect will not be lost. Such compensation must ensure, of course, that firms are not free-riding on the efforts of the FOSS community and that the FOSS project leader cannot appropriate the compensation. Details of and possible obstacles to setting such a compensation scheme are discussed below.

One can also question the need to use strict virality to spread FOSS. Rather, as the Apple example indicates, strict and wide-scoped virality might inhibit the spread of viral FOSS, as otherwise it could be used on more platforms and increase its value to potential users. But more importantly, the spread of viral FOSS is based on its quality for the user which, in turn, is affected by the motivations of developers to contribute to it. Accordingly, the effects of the level of virality on such motivations are key to our analysis. We analyze them more carefully.

Let us first focus on volunteer developers. A more flexible virality will likely reduce the motivation of those developers driven by an ideological opposition to commercial software.[63] Yet this is not always true. If developers are motivated by the wish of breaking down the dominant software firms, [64] then moderate virality might better advance this goal, so long as it is applied in a way which takes this consideration into account in weighing whether or not to grant a license. This is because viral FOSS limits the ability and incentives of users who use complementary commercial products to use FOSS. This, in turn, might limit the ability of viral FOSS to reduce the

[63] For such motivations, see, e.g., Guido Hertel, Sven Niedner & Stefanie Herrmann, *Motivation of Software Developers in Open Source Projects: An Internet-Based Survey Of Contributors to the Linux Kernel*, 32 RES. POL'Y 1159 (2003).

[64] Bruce M. Kogut & Anca Metiu, *Open-Source Software Development and Distributed Innovation*, 17 OXFORD REV. ECON. POL'Y 248 (2001).

market power of some dominant firms, thereby going against the basic goal of creating a world of greater progress and less concentration of wealth and corporate power. It is noteworthy that no purely viral FOSS is dominant in its market.

Moreover, as numerous studies on the motivations to contribute to collaborative FOSS have indicated, most developers are not ideologically driven. Indeed, an empirical study conducted by Lakhani and Wolf found that private, intrinsic motivations help induce developers to contribute to open source: developers were primarily motivated by education/intellectual stimulation (29 percent), non-work user needs such as developing better games (27 percent), work-related user needs (25 percent), and feelings of obligation/community (19 percent).[65] The three first motivations, and possibly the fourth, are not harmed by moderate virality. One can still develop FOSS for one's own intrinsic needs, even if it interoperates with commercial products or it serves as a basis for new commercial products.

Furthermore, the possible negative welfare effects of strict virality might limit the motivation of some developers to embed their best ideas in FOSS, for several reasons. First, if one's idea can better increase social welfare if embedded in more moderate viral FOSS or in commercial software, then those seeking satisfaction for furthering welfare might choose not to embed it in viral FOSS. Second, if virality reduces the scope of compatible software, this might reduce its value to its developer-user. Both effects are strengthened by the nature of development of the software industry, in which compatibility and interoperability are important components for creating value. Third, creative pleasure, satisfaction from creation or being part of a team are not lost by moderate virality but rather might be increased in parallel with the scope and quality of the FOSS created. Accordingly, virality might be FOSS' worst enemy to innovation and expansion.

Some researchers have related the motivation of developers to create FOSS to extrinsic motivations such as the reputation they acquire, which is eventually rewarded in the job market.[66] Such a reputation will not be harmed and might be further strengthened if the FOSS they contributed to becomes part of a commercial product, as its commercial success may serve as a (partial) signal of the quality of their contribution. Other studies have shown that peer-recognition is an important factor in creating motivations to

[65] Karim R. Lakhani & Robert G. Wolf, *Why Hackers Do What They Do: Understanding Motivation and Efforts in Free Open Source Projects*, *in* PERSPECTIVES ON FREE AND OPEN SOURCE SOFTWARE, *supra* note 33, at 3.

[66] *See, e.g.*, Josh Lerner & Jean Tirole, *Some Simple Economics of Open Source*, 52 J. INDUS. ECON. 197 (2002); Il-Horn Hann, Jeff Roberts, Sandra Slaughter & Roy Fielding, *Economic Returns to Open Source Participation: A Panel Data Analysis*, Presented at WORKSHOP ON INFORMATION SYSTEMS ECONOMICS (WISE) (2004), *available at* http://opim.wharton.upenn.edu/wise2004/sun412.pdf (finding that a higher ranking within the Apache Project is positively correlated with higher wages).

contribute.[67] Ensuring that code lines carry the name of the developer who wrote them can be done regardless of the level of virality of the code. Accordingly, once again, moderate virality might strengthen such motivations.

These observations are supported by economic studies. A study conducted by Fershtman and Gandall found that the output per contributor in FOSS projects is much higher when licenses are less restrictive and more commercially oriented.[68] Similarly, another study found that the more restrictive the license, the lower the probability that the project will reach an advanced development stage.[69] These findings suggest that more moderate restrictions on licensing can increase the motivations of developers to contribute. Of course, virality need not be abolished completely, and the gate might be opened only to synergies that create significant positive welfare effects.

Let us now turn to the motivations of commercial firms that contribute to FOSS projects. If the firm is motivated by the wish to needle its rivals, virality creates mixed effects. On the one hand, virality might limit the ability of one's rivals to interoperate with the FOSS code, thereby increasing the FOSS' harmful effects. On the other hand, virality might inhibit further growth of the FOSS software due to limitations on possible synergies and concerns regarding its scope of penetrability, thereby reducing its competitive pressures on rivals. The supporting firm might thus prefer a more moderate virality, which allows some synergies, so long as they do not benefit its rivals. If the firm is motivated by the need to overcome host appropriability concerns by way of creating a commoditized platform for its complementary products, again it will generally prefer moderate virality in order to increase the use of its platform and to allow future interoperability of its commercial products with it. Moderate virality can also advance other motivations, such as the creation of a reputation for openness and generosity or ensuring that the FOSS makes best use of one's comparative advantages. Indeed, in the example given above, IBM did not adopt strict virality. Other major commercial firms also chose to release their product as an open source project under less restrictive virality terms (for example, Chromium and Android), thereby increasing synergy. Accordingly, moderate virality generally does not harm, and may even strengthen, the motivations of commercial firms to contribute to FOSS. Interestingly, the compensation received by a FOSS community for enabling synergy may reduce the need for support by commercial firms.

[67] Alexander Hars & Shaosong Ou, *Working for Free? Motivations of Participating in Open Source Projects*, 6 Int'l J. Electronic Com. 25 (2002); Hertel, Niedner & Herrmann, *supra* note 63.

[68] Chaim Fershtman & Neil Gandall, *Open Source Software: Motivation and Restrictive Licensing*, 4 Int'l Econ. & Econ. Pol'y 209 (2007). A potential problem with this finding is that it equates contribution to the number of lines written by each developer. Yet this might not correlate to the quality of the contribution.

[69] Stefano Comino, Fabio M. Manenti & Maria L. Parisi, From Planning to Mature: On the Determinants of Open Source Take off (2007), *available at* http://elsa.berkeley.edu/~bhhall/e222spring07_files/CaminoManentiParisi07_OSSSuccess.pdf.

Can one argue that the fact that most FOSS projects are licensed under the strictly-viral GPL can be used as an indication that such licenses reflect the preferences of most developers? Several factors mitigate against such a conclusion. First, some developers might not be aware of the societal cost created by limiting synergies. Second, once the GPL has become a standard license to many major FOSS projects, those that wish to enjoy the benefits that flow from its standardization must also incur its costs without having an option of negotiating its terms. Third, interconnection with existing viral FOSS projects mandates its adoption. Finally, many large-scale FOSS projects are not licensed under the GPL.

The above observation leads to a potential cost of introducing a more moderate virality condition: loss of certainty created by standardization. Indeed, a major incentive to choose the GPL as the legal platform for a FOSS is based on the fact that it has become standard. This, in turn, reduced licensing costs, given that developers and users are aware, beforehand, of the bundle of rights attached to the license.[70] But will introducing a more moderate virality condition necessarily lead to the loss of standardization benefits? Several factors might mitigate the negative effects. Most importantly, the GPL has been changed twice before and has not lost its status as standard.[71] Furthermore, if a more moderate virality will increase the incentives of developers to contribute to the FOSS and will increase its diffusion, then incentives to adopt it will be strengthened, as will its use as standard. In addition, the certainty created by standardization will not be significantly harmed as long as the change is not major—such that synergies will be allowed only in a narrow subset of cases that significantly increase welfare—and a standard form of a moderate virality condition will be agreed upon.

Yet, even if we assume that a more flexible virality condition would not significantly affect the incentives of developers to contribute to the creation of FOSS, we should also verify that it does not significantly reduce the incentives of commercial firms seeking synergy and interoperability to innovate, thereby harming dynamic efficiency. Several factors and tools might be used to reduce such a concern. First, exceptions to virality should be limited to those cases in which clear and long-term advantages to social welfare from synergy and interoperability will be created, which eclipse the procompetitive effects exerted by virality. Second, compensation should ensure that there is no free riding. These conditions are further elaborated below.

IV. INDUSTRY REACTIONS AND POSSIBLE (PARTIAL) SOLUTIONS

The limitations on synergies arising from strict virality have spurred the market to seek ways to overcome them. Accordingly, this part analyzes self-

[70] Elkin-Koren, *supra* note 14.
[71] Challenges in changing the GPL are analyzed below.

help tools, both technological and changes to licensing terms, to check whether the market can correct itself. Given our conclusion that self-help remedies are limited, it then analyzes legal tools that might be applied to challenge the scope of virality. All tools are evaluated in light of the tradeoff identified above.

A. Self-help Technological Solutions

1. "Write Around" the Source Code

Copyright law protects expressions but does not protect ideas.[72] Accordingly, if a good idea can be embedded in a different source code, without directly copying the specific code, then this would not infringe copyright. Indeed, the fact that FOSS code is open makes it easier to understand how the code works and to "write around" it, in order to create code that achieves similar results. This implies that good ideas could possibly be used elsewhere. Notably, allowing such "writing around" is part of the balance created in copyright law between creating incentives to innovate and ensuring dynamic growth in a society which builds on new and innovative ideas.[73]

Yet three technological obstacles limit the ability to "write around" some viral FOSS. First, the FOSS might perform the task much more efficiently than a "write around" code. Second, recreating the code might be lengthy and costly. The third barrier might be the most difficult to overcome. A "write around" code that only uses part of the FOSS code will lose other benefits of the FOSS, such as frequent updates, interoperability with the integrated environment in which the FOSS operates, or its large customer base. Accordingly, if the idea is best used as part of a wider network, the competitor might have to develop his own integrated system from scratch. This obstacle to the realization of synergies increases as the quality and scope of code written under viral FOSS is increased. The difficulties in "writing-around" FOSS code are reflected, *inter alia*, in the fact that commercial firms sometimes buy non-GPL licensed FOSS firms and commercialize their products. In addition, this solution does not solve the obstacles faced by firms wishing to operate only in a complementary-product market.

2. If You Can't Beat Them, Join Them

Another market reaction to viral FOSS involves harnessing its benefits to strategic objectives. Above we analyzed several ways to use FOSS to advance commercial firms' goals, including needling one's competitors and ensuring that the FOSS makes best use of your comparative advantages in a

[72] Section 102(b) of the Copyright Act 17 U.S.C. §§ 101-810 (1976).

[73] Whether the open nature of the code changes this balance by making writing around easier is an important question that deserves an article of its own.

complementary market. Yet, as noted, strict virality might limit the achievement of some strategic goals, including the production of welfare-increasing synergetic products.

3. Require the User to Make the Linkage

In order to circumvent virality, some firms sell their software without the viral FOSS component. The user is then required to download it from the web and make the linkage. The wording of the GPL does not directly cover this conduct, as it possibly comes under use of the software, which is allowed. However, as elaborated below, some FSF representatives argue that the purpose and the spirit of the GPL cover such conduct. This question has not yet been settled. Yet, even if it is legally allowed, this technological solution is effective only for synergies that require interoperability, and not synergies that are embedded in new products, and only where such an external linkage does not significantly harm the performance of the commercial software.

4. Create a Technological Buffer

Another way that firms employ to circumvent virality is to design the interface between the FOSS and the commercial code through a technological buffer, instead of a link. For example, they create a software that will call the FOSS, upon the use of the software by the user, to perform a task, and then they use the output in the commercial software. Under the common interpretation of the GPL in the industry, only the code of the "calling software" is considered a derivative work and thus such a use of FOSS does not infringe the GPL. This solution suffers from all the limitations of the previous one.

B. Self-Help Solutions: Changing Licensing Terms

1. Dual Licenses

Some FOSS communities, such as MySQL, have created a dual system of licenses for the use of their code, one with virality and one that allows use for fees. Consequently, users who wish to incorporate the FOSS in an open source application can do so under the GPL. Those wishing to distribute it as part of a commercial product can purchase a commercial license. Such duality is facilitated by the public good nature of software, which enables the software to be used simultaneously by many users, with a different bundle of legal rights attached.

This solution has important positive social welfare effects. While still benefitting the community of FOSS users, it solves the synergy problem with commercial firms. Importantly, commercial entities will realize only those synergies that cannot be efficiently realized by the FOSS community, as otherwise the value of the code to the commercial firm will be low,

thereby creating an efficient allocation of synergy projects. Interestingly, the fact that the product is also available in FOSS form can serve the licensor commercially, as it can enlarge the number of users, thereby increasing the awareness of potential buyers to its software. In addition, dual licensing creates a more financially viable environment for the FOSS project, thereby strengthening the possibility for its continued operation and for its ability to provide services that will further strengthen FOSS, such as financing a supervisory body that will ensure that only efficient changes to the FOSS code are embedded.[74] Some firms adopt a strategy under which the code available in FOSS has more limited features than the code that can be bought, thereby enabling users to experience and experiment with the code and strengthening motivations to buy the enhanced code.

Yet the application of this solution in practice to overcome synergy obstacles is quite limited. First, it is only available to developers of an original code who have full ownership of the code. If any part of the code is already subject to virality, the dual licensing option is not available. Second, since software is constantly evolving and requires ongoing updates, this solution is only short term. Further developments of the code will emerge in two parallels that would not benefit one another. Modifications by commercial parties would not be freely available to FOSS users, and modifications created in the viral FOSS would not be available for incorporation in synergetic products. Accordingly, second-generation synergy is limited.

2. Creation and Use of More Flexible Licenses

The GPL is not the only type of license available for FOSS, although it is commonly used. Indeed, the Open Source Initiative (OSI), a body created in the late nineties as a counterpoint to FSF, which is less ideological and a more business-friendly voice for FOSS, seeking to combine the open source development methodology with some of the benefits commercial markets have to offer, has certified more than sixty licenses as conforming to their Open Source Definition, which verifies that the license embodies FOSS principles. Such principles mandate that the license be open and free, but do not require virality.[75] For example, the OSI certified the Mozilla Public License, which enables the commercial distribution of a combination of FOSS and commercial software, without conditioning it on virality. Additional flexible licenses, not certified by the OSI, have also emerged.[76] Such licenses are indeed used in many new projects. Google, for example, whose business model is based on advertisements rather than payment for

[74] Vetter, *supra* note 9.

[75] Open Source Initiative, http://www.opensource.org/docs/definition.php (last visited Dec. 4, 2011).

[76] Robert W. Gomulkiewicz, *Open Source and Proprietary Models of Innovation: Beyond Ideology: PART III: Open Source and Proprietary Software Development: Open Source License Proliferation: Helpful Diversity or Hopeless Confusion?*, 30 WASH. U. J.L. & POL'Y 261 (2009).

services, operates with Android, which is FOSS. Yet it chose to use a more moderate license (Apache), as the GPL would have potentially limited the evolution of the software ecosystem by discouraging commercial development on top of the FOSS platform. What provoked this outpouring of licenses? The main motivation involves the inability of the GPL to allow cooperation and sharing with commercial software.

Yet, such new licenses do not solve the obstacles to creating synergies to existing viral FOSS. In addition, license diversity carries costs. Indeed, the OSI has identified the issue as one of its most strategic matters.[77] The loss of standardization increases the information costs of both the user and the developer, who must understand the intricacies of and choose among the different licenses available.[78] Indeed, the continued domination of the GPL license might be attributed, at least in part, to the wish to limit such costs. Another cost, which also derives from the loss of standardization, involves determining the compatibility of license terms for a bundle of FOSS programs, which are protected by a variety of different licenses. The distributor of such a bundle must verify that the licenses permit the code to be combined in the package and must determine which rights and obligations must be passed downstream.[79]

3. Change From Within: Create Exceptions in the GPL

Modifying the GPL, and relatively similar licenses, so that they would enable welfare-enhancing commercial synergies with FOSS code, provides the first-best solution to the synergy problem. Such exceptions should be rare and balance the conflicting forces observed above, especially the importance of the synergy for social welfare and the motivations for innovation both of commercial firms and of contributors to FOSS. In addition, the decision whether to grant an exception should take into account the market position of the requesting firm. On the one hand, the FOSS community should be cautious not to strengthen the "cathedrals," the dominant firms in a given market, as they might use FOSS to preserve their competitive edge and their market power. This would harm consumers by using the very product that was created to limit their power. On the other hand, such cathedrals might be best positioned to maximize the social welfare embedded in the new ideas. These competing considerations must be carefully balanced.

Yet applying this solution is fraught with problems. First, it necessitates the creation of a new business model to determine whether a specific case justifies an exemption, and what compensation should be paid for the use of the FOSS code. Changing the terms of use in large scale collaborative

[77] *Id.*
[78] *Id*; Elkin-Koren, *supra* note 14.
[79] Gomulkiewicz, *supra* note 76, at 282.

communities is fraught with difficulties.[80] Formally, this problem can be overcome by the fact that, according to its own terms, the GPL can be amended by the FSF.[81] Yet, changing the license is not simply a formalistic step. Such a change might not be problematic for those who adopted the GPL, not because it is optimal, but because of its quasi-brand identity, espousing certain development procedures or ideological beliefs that developers may find more important than the terms themselves.[82] It might also not be problematic for those developers or users who value the social welfare created by the synergy over other values. Yet, it clashes with the ideology and vision of those members of the FOSS community whose vision and ultimate goal is to create, at any cost, an environment in which all code is FOSS. Giving away some of your best chips to allow synergy might be regarded as a death bed for reaching the ultimate goal, thereby significantly reducing their motivation to contribute to the code. Of course, the more limited the exceptions that could be granted, the less significant the clash between the different groups of developers and users.

Furthermore, some of the benefits that flow from the standardization of the terms of use of the GPL will be lost, although incorporating a standardized and limited version of a moderate virality condition may limit this problem. Moreover, a change that is only applied to licenses granted in the future will create interoperability problems with FOSS licensed under viral versions of the GPL.

Compensation may also be a thorny issue. Distribution of economic profits among developers would generally be problematic, given that this would change the whole motivational structure and might also be open to abuse, as well as practical problems. A better solution is to use such profits to further the specific FOSS software or FOSS in general. For example, funds might be used for marketing, distribution, or even for buying code that would greatly increase the operation of the FOSS by way of synergy. Funds might also be used to strengthen the internal supervision of the creation of the FOSS, to determine which parts of added code increase its

[80] Elkin-Koren, *supra* note 14. Some of these difficulties are exemplified by the objection of many in developers to a patent-sharing agreement between Microsoft and Novell that enabled Microsoft, in accordance with GPL*v*2, to use some of Novell's patents that were also used in Linux in its closed source, in order to create interoperability with Linux. One of the criticisms was that such agreements were not needed in order to create interoperability. This led to the adoption of GPL*v*3.

[81] GNU GENERAL PUBLIC LICENSE, VERSION 2, § 9; GNU GENERAL PUBLIC LICENSE, VERSION 3, § 14 (Free Software Foundation, June 2007). Both sections state that such a revision "will be similar in spirit to the present version but may differ in detail to address new problems or concerns." One can question whether changes in the strength of the virality condition fall within this condition. Nonetheless, this does not prevent the creation of a GLPv4.

[82] David McGowan, *SCO What? Rhetoric, Law, and the Future of F/OSS Production* 33-34 (Univ. of Minn. Law School, Research Paper No. 04-9, June 12, 2004).

value to users and which parts do not. Indeed, such compensation can solve the funding issues of FOSS projects.

Finally, to apply this solution, all the copyrights in the code should belong to the entity making the exception. This might be difficult where code contributors did not assign copyright to the FOSS project, and, thus, parts of the code cannot be assigned to others without engaging in costly negotiations and transactions with code developers.[83] Accordingly, a code repository would need to be built to ensure that rights of use can indeed be transferred to others under less viral terms.

C. Legal Solutions: Challenging Viral Licenses

Given the limited ability of self-help tools to overcome the obstacles to welfare-enhancing synergies created by virality, let us now turn to possible legal tools that might overcome such obstacles. Like all other licensing restrictions, viral restrictions are subject to scrutiny under a number of statutes and legal theories, including copyright, antitrust, and laws against unfair contractual terms. This section briefly explores the first two, which are most relevant. Indeed, the fact that viral FOSS makes use of legal tools (most importantly copyright law) to apply its virality strengthens the need to ensure that such use does not clash with the goals that such legal tools were set to achieve.

1. Challenging the Scope of Virality Under Copyright Law

The GPL requires that all "derivative works" or "work[s] based on" the code also be licensed under its terms of use. Accordingly, an important issue involves the definition of such works.[84] For example, if a software only creates a reference to a viral library that would be activated in run-time and does not incorporate any part of the viral code in its own code, will it be considered a work covered by the GPL? Or if the proprietary software only exchanges information with the GPL'd software, does this fall under the GPL? The scope of the GPL and its virality are crucial, as nearly all programs today include external components, such as libraries, interfaces and plug-ins, and interact or interoperate with other programs.

While the FSF generally adopts a wide interpretation of the GPL, some scholars, firms, and organizations have challenged this view and have argued for a narrower scope.[85] Such an interpretation would allow for more synergy, although it would not solve the problem completely. The relevant

[83] *See also* Barnett, *supra* note 2, at 1898.

[84] *See, e.g.*, Malcolm Bain, *Software Interactions and the GNU General Public License*, 2 IFOSS L. REV. 165 (2010).

[85] For example, the Freedom Task Force created a Software Interactions Document, the aim of which is to provide some general guidance to lawyers and developers working with free software.

legal analysis regarding the scope of the GPL involves two interrelated questions.

The first is legal: if the works covered under the GPL are wider than those protected by copyright and thus restrict otherwise unrestricted acts, what takes precedence. In other words, does the creator of a code have control over his work which extends beyond that granted by copyright?[86] The answer to this question is of much importance, since under the FSF's interpretations, the GPL magnifies the rights of the code author beyond those granted by copyright, *inter alia*, by adopting a wide definition to derivative works.[87] This interpretation is at odds with the narrow definition given by U.S. courts to derivative works in copyright law.

The answer is partly based on whether the GPL is a license or a contract. A license cannot provide a right of use to anything that under copyright belongs in the public sphere.[88] Contract law is more flexible: since the work was created by the author, it might be argued that he should be allowed to determine on what terms he would like to offer it for use by others, subject to agreement by the other party. In our view, this distinction has an inherent flaw that results from the fact that copyrights incorporate an inherent balance between the state-granted exclusivity, which strengthens motivations for creativity, with the creation of a public domain for unprotected works that would facilitate further creativity. This balance rests on public policy.[89] To put it differently, copyright is a "deal" between the author and the state, which draws the boundaries of exclusivity in accordance with public policy. Therefore, by privately extending his rights beyond those granted by copyright, the author is overthrowing the delicate balance reached with regard to the use of information in society.[90] Thus understood, any author restricting the use of works in the public domain is also limiting what he does not own and harming public policy.

To ensure that the above argument should also apply in the case of FOSS, one should also check whether FOSS should be treated differently than other creative works, including commercial code, since the "deal" reached in copyright law is not sufficient to motivate its creation, owing to its unique characteristics. The most relevant characteristic is the open nature of FOSS, which makes it easier for others to observe and write around the

[86] Elkin-Koren, *supra* note 14.

[87] *See also* Bond, *supra* note 35.

[88] In *Jacobsen v. Katzer*, the court found that restrictions on redistribution of FOSS which related to notices on the source of the code were necessary to accomplish the objectives of the open source licensing collaboration, including the enjoyment of its economic benefits, and were protected under copyright. Yet there the restrictions did not prevent the ability to commercially use the code or to create synergies with it. 535 F.3d 1373 (Fed. Cir. 2008).

[89] *See, e.g.*, Kenneth Dam, *Some Economic Considerations in the Intellectual Property Protection of Software*, 24 J. LEGAL STUD. 333 (1995).

[90] Elkin-Koren, *supra* note 14.

code and might thus limit incentives to create the code in the first place. To make such an argument, one would first need to prove that the motivations for the creation of FOSS are even lower than those of a commercial firm operating under similar copyright protection boundaries. Such proof might not be easy and depends, *inter alia*, on what motivates developers of FOSS. Yet, even if it could be proven, the effect of the boundaries set by copyright law on incentives to create FOSS would have to be balanced against the synergy argument. Any limitation imposed on the scope of works in the public domain limits possible synergies between different products. Furthermore, even if FOSS indeed justified a different balance than that provided by copyright, this would not imply that the change of such boundaries should be done through private contracting, rather than though a legislative process that would ensure that all public policy concerns are addressed. Indeed, as Elkin-Koren has elaborated, due to externalities, private ordering cannot be relied upon to protect the public interest.[91] Moreover, even if externalities were absent, the bargaining position of the two sides might affect the outcome, once again not reaching the optimal outcome for society.

Consequently, in our view, the GPL cannot enlarge the scope of covered works beyond that which is granted by copyright. If this view is accepted, there is no need to analyze the second question posed below, which relates to the GPL's scope, unless the GPL is narrower than copyright. Nonetheless, since the above question was not, as of yet, answered by courts, the analysis below continues under the assumption that the GPL can extend the rights granted to the viral FOSS owner beyond those granted by copyright.

Accordingly, the second question is factual: what is covered by the GPL? An important source is, of course, the wording of the GPL itself. Yet the GPL has multiple and sometimes conflicting definitions regarding its scope. Additional sources are the guidelines and answers published by the FSF,[92] especially those which were published before a GPL license was adopted. Finally, since the GPL contains some terms of art, their interpretation by legislators and courts may also be relevant.

There seems to be agreement on some types of interactions that come under the GPL, such as statically linked programs, where the GPLed code constitutes an integral part of the new software. There also seems to be agreement that certain types of interactions are not covered by the GPL. For example, pipes, sockets, and command-line arguments, which are communication mechanisms normally used between two separate programs, generally do not come under its scope.[93] Yet there is no agreement on many other types of interactions.

[91] *Id.*
[92] Free Software Foundation (FSF), http://www.fsf.org/ (last visited Dec. 4, 2011).
[93] *Id.*

One of the most important and hotly debated issues is whether dynamic linking is covered by the GPL. Dynamic linking occurs when the content of external programs or library files ("external programs") is not copied into the new software, but instead the software refers to those external programs in its code. When the software is run by its user, the external programs may be "called up" and executed. This means that neither the source code of the software nor the compiled and linked executable of that software contain the source code of the external program. Although the external program is linked at run-time, this is only created in the user's computer memory. The answer to this question is crucial for synergy based on interoperability, which is an important way in which software markets evolve.

Some have argued that since the external program is only reproduced and distributed at run time, it is not covered by the GPL. On the other hand, the FSF applies a wide view, which rests on two arguments.[94] First, the plug-in to the external program is an integral part of the new software, and thus the new software is not an "independent and separate" work in itself. Second, the new software is interdependent on the GPLed code, since it was designed and written to include and use the functionalities of the external program. Bain raises some counter-arguments: writing code to use an external program could be considered merely "using" it in the intended manner covered by the GPL. Moreover, if the program can run any other external program with similar functionalities, the dependency argument is much weaker. These questions have yet to be determined by courts.[95]

Yet, even if these legal and factual interpretations that narrow the scope of the GPL and enlarge the ability to create synergies are correct, some features make it difficult for firms to actually challenge the scope of the GPL. First, there is no clear-cut answer in copyright as to what constitutes a "derivative work." This results, *inter alia*, from the fact that copyright laws were not created for software, and thus often produce unclear results with regard to the scope of their application. Second, given that disputes are based on technology-intensive facts, there is an inherent risk of indeterminacy in characterizing the underlying facts.[96] Third, while the use of technology is not restricted by national boundaries, copyright is a national right. Accordingly, the scope of copyright protection may vary among jurisdictions in which the software licensed under the GPL is used. Another complication is created if the GPL is considered a contractual document, to which varying jurisdiction-specific rules of contractual interpretation may apply.[97] Accordingly, as the number of jurisdictions in which a commercial software

[94] Bain, *supra* note 84.
[95] *Id.*
[96] Melville B. Nimmer & David Nimmer, Nimmer On Copyright § 13.03(A)(1)(d), 13-51 (Matthew Bender 2003) ("Software developers have no adequate guidelines regarding what level of independent development is required to avoid copyright infringement.").
[97] Bain, *supra* note 84.

is used increases, so does the risk of a GPL infringement, as well as the costs of legally determining the GPL's scope.

A fourth obstacle involves the height of the penalty. The infringer would most likely be mandated to withdraw the infringing product and damages might also be granted. As a result, creative and distributive efforts might be lost. Interestingly, in the few cases brought by the FSF, which were settled out of court, the agreed upon sums were not disclosed. This creates a continued vagueness as to the extent of risk involved in such infringements. Moreover, and sometimes even more importantly, the penalty for such a challenge is not merely legal. Rather, an important effect is reputational. The GPL is backed by a community which invests effort in detecting possible violations and strongly voicing its opposition.[98] Such peer pressure uses two primary techniques: chastisement and denigration. Accordingly, the battle over legal interpretations is also fought outside the courts.

Finally, while the FSF acts as a community with shared interests, private parties face a collective action problem, where each firm, on its own, would not bear the costs and risks of challenging the interpretation of the FSF to the GPL, although it would have such a motivation if all firms joined forces and shared costs and risks. Accordingly, firms are reluctant to be the first to take the hard cases to court. Interestingly, the FSF has not brought hard cases, which could have clarified such issues, to court and have instead focused on easy cases with outright infringements.[99] This might be an intended strategy, aimed at preserving the high costs of challenging the validity of their interpretation. The bottom line is that legal and practical obstacles limit the use of copyright law to overcome the synergy problem.

We conclude this part by noting that, unlike patent law, copyright law does not provide for a possibility of compulsory licenses. Yet, given the economic role of software, which is often more similar to patented inventions than to traditional copyrighted works such as books and plays, the rationales that justify compulsory licensing of patents might also be relevant to copyrighted software. The copyright misuse doctrine, which applies if the copyright owner extends his rights in a manner which violates the public policy embodied in the grant of the copyright, does not solve this deficiency.[100] It is a limited tool, *inter alia*, since it can only be used as a shield against infringement suits brought by the copyright owner and not as a basis for a claim to narrow the GPL's virality. Tzur and David have suggested

[98] *See, e.g.*, THE GPL-VIOLATIONS.ORG, http://gpl-violations.org (last visited Dec. 4, 2011), which works together with the FSF to search for violations.

[99] For example, the cases brought against Verison or Busybox were straightforward infringements where GPL code was simply copied and embedded in commercial code.

[100] *See, e.g.*, Christian H. Nadan, *Open Source Licensing: Virus or Virtue?*, 10 TEX. INTELL. PROP. L.J. 349, 368 (2002); Robin Feldman, *The Open Source Biotechnology Movement: Is It Patent Misuse?*, 6 MINN. J.L. SCI. & TECH. 117, 118 (2004); Vetter, *supra* note 2.

establishing a fair use exception to welfare-increasing synergies,[101] yet this suggestion requires significant changes in copyrights.

2. Compulsory Licenses: Virality as an Antitrust Violation?

An interesting question is whether antitrust places limitations on viral FOSS and, if it does, are the existing rules fit to the task, or does dealing with viral FOSS require new thinking in established antitrust doctrines?[102] The question arises, *inter alia*, because antitrust doctrines were designed to apply in commercial markets, whereas the social production model is not based on monetary profit.

Let us first focus on the prohibitions against agreements in restraint of trade.[103] This requires us first to determine whether the GPL is a horizontal or a vertical agreement. Where a software is developed by a community of developers, it has elements of a horizontal agreement. Yet socially-produced FOSS challenges this view. This is because developers are usually not potential rivals, but rather parties to a joint collaborative effort that would not otherwise compete with each other.[104] The more relevant relationship is vertical, as the GPL constitutes an arrangement between the owner of the FOSS and the user. Yet FOSS often challenges this relationship as well, given that many developers are also users. This quasi-vertical relationship implies that virality should be analyzed under a rule of reason, under which anticompetitive effects are balanced with procompetitive ones.[105]

The more difficult issue is whether virality creates an antitrust injury.[106] As elaborated above, virality creates complex effects on both competition and innovation. On the one hand, it limits potential synergies between FOSS and commercial code which could have increased dynamic efficiency.

[101] Tsur & David, *supra* note 40.

[102] Predatory pricing serves as one example, where established doctrines require recoupment of costs, a condition which is not relevant when the product is distributed for free. *See, e.g.*, Wallace v. Int'l Bus. Mach. Corp., 467 F.3d 1104 (7th Cir. 2006).

[103] Section 1 of the Sherman Act, ch. 647, 26 Stat. 209 (1890), 15 U.S.C. § 1 (1934); Section 5 of the Federal Trade Commission Act, ch. 11, § 5, 38 Stat. 719, 15 U.S.C. § 45 (1914).

[104] Note that developers generally do not compete among themselves in the software market, although they may compete for the provision of software development services for firms which compete in software markets. Exceptions may arise, for example, when competitors join efforts to create a FOSS. *See, e.g.*, Maurer, *supra* note 9.

[105] United States v. Trenton Potteries Co., 273 U.S. 392 (1927). In *Wallace*, the U.S. Court of Appeals for the Seventh Circuit found that the GPL does not restrain trade. Rather, it is a cooperative agreement that facilitates production of new derivative works, and as such is lawful. Yet the focus of the court was not on virality, but rather of the fact that the code was distributed for free. The court's analysis thus leaves the door open for future analysis of claims that certain GPL conditions pose a threat to welfare in the long run. *Wallace*, 467 F.3d at 1104.

[106] Anticompetitive effect has been described as a reduction of output, increase in price, or deterioration in quality of goods and services. *See, e.g.*, Generac Corp. v. Caterpillar Inc., 172 F.3d 971, 978 (7th Cir. 1999); United States v. Brown Univ., 5 F.3d 658, 668 (3d Cir. 1993).

On the other hand, it strengthens the motivations of commercial firms to compete with FOSS. Virality also creates complex effects on the motivations of developers to contribute to the production of viral FOSS. Given that the relative size of such effects is not easily quantifiable, and that interfering in the existing relationship amounts to second-guessing software designers' architectural and motivational choices, a hands-off approach should generally be adopted. Only in extreme cases, in which the effects on long-term consumer welfare are distinctly negative, should courts intervene and apply a less viral alternative that would ensure that significant welfare-enhancing effects are realized.

Where the viral FOSS enjoys a monopoly position, refusal to deal prohibitions might also be relevant if the refusal extends, preserves, or creates significant market power.[107] Most relevantly, the monopolist might incur antitrust liability under the essential facilities doctrine.[108] Under the doctrine, if the product or service supplied by the monopolist is an essential input for operating in a (connected) market—that is, it cannot reasonably be duplicated by a competitor—and the monopolist does not have an objective justification for refusing to supply it, then he will be required to supply it on nondiscriminatory terms.[109] To apply to viral FOSS, no write-arounds or technological solutions that may be costlier but would still allow firms to operate economically in the market to which access is sought should be possible. While the doctrine has suffered severe criticism and its scope has been significantly reduced, the case for its application may be strengthened where the viral FOSS is the basis of a software infrastructure. This is because virality creates an extremely high obstacle to operating in a complementary

[107] In CSU v. Xerox, the Court limited antitrust liability for refusals to deal involving intellectual property only to cases involving illegal tying, fraud, or sham litigation. 203 F.3d 1322 (REd. Cir. 2000). Contrast this decision with United States v. Microsoft Corp., 253 F.3d 34, 58-59 (D.C. Cir. 2001). The CSU decision was criticism by many commentators and is not supported by the Department of Justice.

[108] Closely related is antitrust liability for dominant firms who refuse to deal with competitors. *See, e.g.*, Richard J. Gilbert & Carl Shapiro, *An Economic Analysis of Unilateral Refusals to License Intellectual Property*, 93 PROC. NAT'L ACAD. SCI. USA 12749 (1996); Michael A. Carrier, *Refusals to License Intellectual Property After Trinko*, 55 DEPAUL L. REV. 1191 (2006); David McGowan, *Regulating Competition in the Information Age: Computer Software as an Essential Facility Under the Sherman Act*, 18 HASTINGS COMM. & ENT. L.J. 771 (1996).

[109] *See, e.g.*, MCI Commc'ns Corp. v. Am. Tel. & Tel. Co., 708 F.2d 1081, 1132-33 (7th Cir. 1983), *cert. denied*, 464 U.S. 891 (1983); Robert Pitofsky, Donna Patterson & Jonathan Hooks, *The Essential Facilities Doctrine Under U.S. Antitrust Law*, 70 ANTITRUST L.J. 443, 445 (2002); Brett Frischmann & Spencer Weber Waller, *Reinvigorating the Essential Facilities Doctrine*, 75 ANTITRUST L.J. 1 (2008). The Doctrine has come under severe criticism. *See, e.g.*, Philip Areeda, *Essential Facilities: An Epithet in Need of Limiting Principles*, 58 ANTITRUST L.J. 841 (1990); PHILLIP E. AREEDA & HERBERT HOVENKAMP, ANTITRUST LAW 771c (2d ed., 2002); Verizon Commc'ns Inc. v. Law Offices of Curtis V. Trinko, LLP, 540 U.S. 398 (2004).

market and creating synergetic goods, since the price of access is giving up the ability to commercially profit from the synergetic good. These costs go beyond those which usually exist in infrastructure markets, which include, as Waller and Frischmann elaborate, information and transaction costs as well as positive externalities that increase social welfare yet are not captured by the transacting parties.[110] Accordingly, the social costs of restricting access to such an infrastructure can be significant and yet evade self-regulation within conventional economic transactions.

An additional argument in favor of applying the essential facilities doctrine to viral FOSS can be raised. The application of the doctrine in a traditional commercial setting prevents the incumbent firm from unlawfully acquiring or maintaining a monopoly in the market in which it faces competition, but allows it to profit from its monopoly in its main market by charging (non-discriminatory) access fees to its facility. This incentive structure does not carry over to viral FOSS, given that the motivation to create it is not based on the ability to sell FOSS for profit, and thus access will not be granted in a larger set of welfare-reducing case.[111] A reciprocal argument can also be raised: why should viral FOSS be exempt from legal limitations imposed on proprietary software firms?

Yet the unique characteristics of viral FOSS raise several obstacles to applying the doctrine to it. Most importantly, the court would need to determine whether the ideology behind some viral FOSS developers—that eventually all code should be FOSS even if the price is preventing some synergies—serves as an objective justification for a refusal to deal.[112] The analysis thus involves not only purely economic motives, as is generally the case, but also ideological ones which, in turn, affect innovation. A possible counter argument can be based on the fact that purely ideological motivations play a strong motivational role only in a small group of FOSS developers. An exemption based on such a justification is especially problematic where FOSS is supported by commercial software firms.

Another obstacle relates to the requirement, applied by many courts and accepted by most scholars, that the monopolist should also be a competitor in the market to which access is denied.[113] In a commercial market setting this condition is generally justified. That is, if the monopolist does not

[110] Frischmann & Waller, *supra* note 109.

[111] Mark A. Lemley & Brett M. Frischmann, *Spillovers*, 100 COLUM. L. REV. 101 (2006).

[112] Areeda and Hovenkamp argue that refusals for non-economic reasons are outside the scope of the antitrust laws. AREEDA & HOVENKAMP, *supra* note 109, at Vol. IIIA, § 770, part 7D-3. We argue that such a categorical immunity from antitrust liability is overboard, at least in the case of FOSS. It would enable firms to justify socially harmful conduct on non-market ideologies and would prevent the balancing of interests to ensure that overall social welfare is not harmed. Moreover, FOSS is often created with the goal of challenging existing competitors.

[113] *See, e.g.*, Integraph Corp. v. Intel Corp., 195 F.3d 1346 (Fed. Cir. 1999).

operate in the vertical market, then he will have incentives to allow access to its facility in order to maximize his profits, so long as he is not constrained from charging the monopoly price. In such circumstances, it has been argued, "it is difficult to see how denying a facility to one who . . . is not an actual or potential competitor would enhance or reinforce the monopolist's market power."[114] This creates a presumption that the monopolist's refusal of access is based on legitimate business reasons. Yet this rationale does not carry over to FOSS. Rather, because of its unique nature, and because of the incentives for, and the ease of, creating interoperability between viral FOSS projects, incentives for vertical integration with other viral FOSS projects exist without need of formal ownership, and obstacles to creating synergies with commercial firms are high—even if such synergies are highly profitable and formal vertical integration is absent. In such a situation, it cannot be assumed that denying access to commercial firms will not strengthen the market power of the FOSS firms or that it will not harm consumers. A case can thus be made for dropping the requirement of vertical integration in the case of viral FOSS.

An additional obstacle is remedial. Even if all the conditions of the doctrine are proven, the ability to actually use the code will ultimately depend on the remedy. Usually, the remedy consists of granting competitors access to infrastructure on terms no less favorable than those granted to the incumbent's current internal or external users. In most cases this would allow as-efficient competitors to access and compete in the market. In the case of viral FOSS, however, such a remedy would not change the existing situation. This is because the GPL already allows the use of the code to anyone who is willing to meet its conditions. The court would thus need to determine the terms of trade, a task for which it is ill-fitted. Still, the question is whether the costs of non-intervention are not high enough to justify the risk of error. The fact that the terms of the compulsory license are considered *ex post* reduces such a risk, as it allows the court to fine-tune the terms to take into account the effects on competition and innovation in light of the specific market conditions.

To sum, a case can be made for challenging viraity under antitrust laws, albeit only in extreme circumstances. Yet such an application faces substantial conceptual difficulties that require courts to modify precedents, where needed, in order to take into account the unique features of viral FOSS. Furthermore, many of the practical obstacles that face firms wishing to challenge virality under copyright laws here as well. Given, however, the limitations on self-help, a case can be made for applying antitrust tools to enable synergies which hold the potential to significantly increase social welfare.

[114] Interface Group v. Massachusetts Port Auth., 816 F.2d 9, 12 (1st Cir. 1987).

D. Public Policy Solutions: Public Funding for Synergies

In theory, the fact that a synergy can potentially increase social welfare but cannot be realized due to viral limitations might justify the use of public funding to buy the commercial code needed to create the synergetic good and provide it under viral limitations. This is a public version of "if you can't beat them, join them." Yet such a solution would meet significant practical obstacles. To name a few, a public committee that would need to evaluate the quality of possible synergies will be subject to informational obstacles as well as to pressures from interest groups. In addition, given that the positive welfare effects of the synergy might encompass more than one jurisdiction, there is a risk of free riding, unless all affected jurisdictions finance the synergy.

V. CONCLUSION

FOSS has emerged as a major cultural and economic phenomenon. The paradigmatic shift in the ideology regarding the openness of the source code and the freedom to modify it was accompanied and facilitated by a change in production models, towards collaborative and often voluntary peer production.[115] These changes brought about significant benefits to social welfare, resulting, *inter alia*, from increased innovation and procompetitive pressures in many software markets.

The GPL has emerged as the main legal tool used to facilitate noncommercial FOSS projects. It incorporates several public domain licensing features. Most importantly, it grants anyone the freedom to use, distribute, and modify the FOSS. It also includes a unique condition, a viral term, which requires that any work based on the code or derived from it must be licensed under the GPL as well, thereby granting others the right to use it. This virality condition serves several important purposes. It limits the ability of commercial firms to appropriate the source code; it strengthens the incentives of some developers to contribute to FOSS; and it contributes to the spreading of FOSS.

At the same time, however, it creates potentially negative welfare effects. Its extremeness, which allows no exceptions to its virality, creates (at least) two separate and competing spheres: viral FOSS and commercial software. Competition between the different spheres carries many advantages, since it creates motivations to build "the better mouse trap." Yet by completely separating both spheres, virality prevents some possible software synergies, thereby limiting innovation and dynamic efficiency. Accordingly, the extremity of the virality condition and the wide interpretation given to it by the FSF may, in some cases, be social welfare reducing. In addition, it does not

[115] Eric von Hippel, Democratizing Innovation 99 (MIT Press 2005).

necessarily serve the FOSS movement, as it blocks some important ways for its growth and for FOSS to enhance social welfare. In fact, virality might be viral FOSS' worst enemy. As argued in this paper, a more moderate virality might well be justified.

In light of our conclusion, technological and legal responses to virality were analyzed. As shown, most market responses are partial and do not solve synergy limitations. Only the relaxation of the virality condition by the market or one coerced by courts, or a significant change in virality's adoption patterns in FOSS projects—possibly facilitated by the understanding of the effects of virality's extremity on social welfare—can bring about the needed change. Until then, virality will continue to serve as an obstacle to the vision of a hybrid economy, in which integration of peer production and commercial entities will become the dominant architecture for commerce on the internet, incentivizing future innovation.[116]

[116] Lawrence Lessig, Remix: Letting Art and Commerce Thrive in the Hybrid Economy (Penguin Press 2008).