

A License to Kill (Innovation):
On Open Source Licenses and their implications for Innovation
By Michal Tsur & Shay David¹

¹ Michal.tsur@yale.edu, and sd256@cornell.edu.

Michal Tsur is a fellow at the Information Society Project at Yale Law School, and Shay David belongs to the Science and Technology Studies Department / Information Science Program at Cornell.

Table of contents

Table of contents	2
Abstract	3
1 Introduction	4
2 A (Very) Brief History of Open Source Licenses	7
3 Innovation in Software	14
3.1 Several Useful Distinctions: Developers Versus Users; Initial Developers Versus Later Developers	14
3.2 The Sources of Innovation in software	14
4 Software Licenses and Innovation	18
4.1 The effect that licenses and the availability of licensing schemes may have on innovation	18
5 OSLs and innovation	20
5.1 What type of a license is an open source license?	23
5.2 OSLs and Current vs. Future Innovation	25
5.3 Aspects Of OSLs That Positively Impact Innovation	26
5.4 Potential Negative Effects of Certain OSL Provisions on Innovation	31
6 Drafting OSLs That Maximize Innovation	35
6.1 Is Innovation A Worthy Objective For OSLs (Why Is It Important To Craft OSLs With Innovation In Mind)	35
6.2 Current OSLs Are Not Consistent In Furthering the Goal Of Maximizing Innovation	36
6.3 Suggestions for Improvement	37
7 Conclusion	42

Abstract

This paper examines the implications of Open Source License (OSL) selection on software innovation, and suggests how modifying the Open Source Definition, or modifying certain provisions in OSLs that have become de-facto standard licenses in open source development, could better accommodate the competing needs and diverse motivations of different would-be software innovators. We make an important distinction between “initial developers” – those developers who decide what license will apply to the code they write, and “later developers” – those developers who subsequently wish to use code that was previously released under a certain OSL (and are therefore affected by license terms selected by initial developers). This distinction facilitates the analysis of the effect OSL provisions have on the development of new “independent” code and, importantly, their effect on any subsequent use of code released under an OSL. The changes we propose could considerably increase the likelihood that a wider variety of developers (including commercial firms) would make use of code released under such revised OSLs, as well as the likelihood that code would be released under OSLs to begin with.

1 Introduction

The objective of this article is to unpack some of the mystique that surrounds Open Source Licenses (OSLs), to explicate the implications of OSL selection on software innovation, and to suggest how modifying the Open Source Definition², or modifying certain provisions in OSLs that have become de-facto standard licenses in open source development, could better accommodate the competing needs and diverse motivations of different would-be software innovators. We make an important distinction between “initial developers” – those developers who decide what license will apply to the code they write, and “later developers” – those developers who subsequently wish to use code that was previously released under a certain OSL (and are therefore affected by license terms selected by initial developers). This distinction facilitates the analysis of the effect OSL provisions have on the development of new “independent” code and, importantly, their effect on any subsequent use of code released under an OSL.

As we know, software, as a field, is arguably the site of some of the most significant innovations over the past 50 years. Software has become an integral part of almost any scientific, economic, and, growingly, social activity that surrounds us and, accordingly, many innovations in these spheres have been the upshot of software innovation. To some degree, such significant innovation has been enabled by the very nature of software: software’s digital character allows fast prototyping, concurrent versioning, contribution from teams across geographic boundaries, and upgrades (usually at a low cost) of small incremental changes that can be incorporated at a later time. Moreover, and perhaps more importantly, software can be replicated with perfect fidelity and without detriment to the original ‘blue-print’ thus allowing rapid, low-cost uptake of

² The Open Source Definition consists of a list of certain minimal conditions that the distribution terms of open source software must comply with. See <http://www.opensource.org/docs/definition.php>

innovative products. However, while such characteristics can explain how software innovation occurs, they do not explain why it occurs. Ostensibly, all these technical properties are facilitators and not incentives to innovation. In contrast, the concept of software as property--intellectual property--that can be developed in the prospect for economic profit is indeed one such explanation for software innovation. Not surprisingly given the growing importance of software in our world, pursuing this model allowed several companies to exploit the economic benefits inherent in software development and establish themselves as some of the largest corporations in the world.

There is, however, a growing group of activists and business people from the free and open source software movement that think that this explanation and development models are mistaken. Organizations such as the Free Software Foundation and the Open Source Initiative were founded to serve the belief that when kept strictly proprietary, innovation in software is actually stifled and that the proper way to encourage innovation is to distribute software with its source code under a license that ensures the users' right to modify the software's underlying code to their needs and to redistribute such modified versions. This heated debate over the importance of source access and licensing terms has been simmering for two decades with both advocates of open source systems and their adversaries highlighting the central role that licensing regimes play in maintaining the fire of software innovation. On the one end there are the supporters of proprietary, "traditional," closed source licenses, which typically forbid access and modification of source-code, and under which users only get access to the compiled versions of the software (aka "click-wrap" licenses). On the other end, there are advocates of the open source licenses, which ensure users' access to the source code, but which dramatically differ in the details of implementation from one another. It is on the licenses on the latter end of this spectrum that our interest in this article is focused.

We emphasize the fact that to the extent that OSLs have been enmeshed in and within social, economical and political advocacy, they have largely not been treated as tools that can bridge the existing gaps between ‘closed’ (commercial, for-profit, proprietary) and ‘open’ (free, open access, community-based) systems. Simply put, existing OSLs have not been crafted with an objective to maximize overall software innovation, whatever the motivation for such innovation may be. More specifically, OSLs have not been designed to maximize the amount of software developed and distributed under OSLs. Instead, the different OSLs have all stressed one or another aspect of the ‘rights’ granted in them, and the advocates of each specific OSL have largely ignored the larger implications for software innovation. We argue that this lack of license standardization is an impediment for innovation. More importantly, and in contrast to the belief of some in the free/open source movement, when considering the private benefit problem, the historical and on-going contribution of for-profit firms to the development of software, and the full range of underlying motivations of software developers it is not clear to us that innovation is best served by fully viral open source licenses like the General Public License (GPL) that has become the de-facto standard used by seventy percent of open source projects. We argue that more permissive licenses are probably more innovation inducing and that carefully crafted licenses can enjoy the benefits inherent in the GPL and yet appeal to commercial enterprises. To understand these claims, we shall start with a (very) brief social history of OSLs and a short analysis of OSL in the context of existing intellectual property laws. We later proceed in making some analytical distinctions between classes of developers that will highlight the importance that licenses in general and open source licenses particularly play in software innovation. On this background we assess the effect that various OSL provisions may have on software innovation, and conclude with a set of recommendations for drafting OSLs that maximize innovation.

2 A (Very) Brief History of Open Source Licenses³

Some two decades ago, Richard Stallman, a Harvard alum and MIT Artificial Intelligence lab veteran started the Free Software Foundation with the intent of revolutionizing the way software is written, distributed and used. Instead of viewing software as property Stallman saw software as a common good that just like any other type of ‘commons’ needs to be preserved for the benefit of the community. Together with FSF co-founder Prof. Eben Moglen, Stallman drafted the GNU general public license (GPL), a very innovative unilateral agreement that makes use of existing copyright law to control the distribution rights of free software and to ensure that free software never becomes non-free. To this extent the GPL reverses the traditional use of copyright (instead of protecting rights, it protects freedom) and for this reason registering software under the GPL is known as ‘copylefting’. We will shortly return to the details of the GPL that make it unique, suffice it to note here that (a) the GPL is based on the well established tradition of standard copyrights, and hinges on the 1980 amendments to the copyright act of 1976 that grants copyright in software⁴ and (b) the GPL was crafted with the intent to maximize a common source code repository of code released under similar conditions, a goal that might be consistent with the goal of maximizing innovation but might not be fully congruent with this latter aim (in other words, while the GPL probably induces innovation, as we argue below, it does not maximize it.)

³ The history of the free and open source movements is detailed in several books and articles, most famous of which is a collection of essays by key people in the free and open source movement, see Chris DiBona, Sam Ockman, and Mark Stone, *OPEN SOURCES VOICES FROM THE OPEN SOURCE REVOLUTION* (Beijing ; Sebastopol: O'Reilly, 1999). The Open Source Organization homepage proposes a history at *History of the Osi* [Web] (Open Source Organization, 1999 [cited 05/04 2004]); available from <http://www.opensource.org/docs/history.php>. The history of Linux is outlined in Glyn Moody, *THE REBEL CODE : THE INSIDE STORY OF LINUX AND THE OPEN SOURCE REVOLUTION* (Cambridge, Mass.: Perseus Pub., 2001). and by Torvalds himself, see Linus Torvalds and David Diamond, *JUST FOR FUN: THE STORY OF AN ACCIDENTAL REVOLUTIONARY* (New York: HarperBusiness, 2001). A good survey of legal issues concerning OSL is given by Dennis M. Kennedy, *A PRIMER ON OPEN SOURCE LICENSING LEGAL ISSUES: COPYRIGHT, COPYLEFT AND COPYFUTURE* [Web] (2001 [cited 09/06/04 2004]); available at <http://www.denniskennedy.com/opensourcedmk.pdf>. There are also two recent books surveying open source licenses: Andrew M. St. Laurent, *UNDERSTANDING OPEN SOURCE AND FREE SOFTWARE LICENSING*, O'Reilly, 2004; and Lawrence Rosen, *OPEN SOURCE LICENSING*, Prentice Hall PTR 2005.

⁴ 17 U.S.C 106, and 17 U.S.C 117.

It is evident that since the conception of the GPL it became almost a de-facto standard whereby the large majority of free software authors chose to release their projects resorting to it⁵. However, in some important senses the GPL has always been too powerful a means to a cause, since it is a fully ‘viral’ license, meaning that any derivative software that makes use of GPLed software, under the conditions set forth in the GPL, should itself be released under a GPL-like license⁶. This creates a problem for for-profit software development entities because the demand to release the source code of derivative work undermines the attempts to sell innovative software for a profit. And indeed several years after the drafting of the GPL new social groups were formed, like the group of hackers piloted by Eric Raymond and Bruce Perens⁷ that decided to create a new framework for open software and information-sharing that would combine the values of free software with the virtues of capitalism. The new group believed that free software has many more merits to offer than just social values and that it should be endorsed by businesses as a superior development methodology and a logical business practice. To ensure that development is continued according to its principles the OSI released the ‘open source definition’, a document that describes the minimum requirements for software to become ‘open source’, together with a new licensing agreement (the Mozilla Public License – MPL) that was much more permissive than the GPL⁸. Specifically, the MPL allowed the amalgamation of open-source software with proprietary software in a commercial software distribution without requiring such derivative work to be released under a free software license, a requirement that is central in

⁵ According to an analysis of data available from SourceForge.net (the largest site for the management of open source projects) the distribution of OSLs in August 2004 was as follows: Public Domain - 3%, BSD License - 7%, Other - 9%, Lesser General Public License (LGPL) - 11%, GNU General Public License (GPL) - 70%. Raw data available at <http://sourceforge.net/softwaremap/trove_list.php?form_cat=14>

⁶ The GNU General Public License § 2. Available at <<http://www.fsf.org/copyleft/gpl.html>>

⁷ See a detailed list of key historical events of the OSI at <http://opensource.planetjava.org/docs/history.php>

⁸ The open source definitions sets 10 conditions that open source licenses need to meet in order to be considered as such. See <<http://www.opensource.org/docs/definition.php>>. The OSI encourages the use of the Mozilla Public license that is available at <<http://www.opensource.org/licenses/mozilla1.1.php>>

the GPL. Relaxing this requirement cleared the way for major businesses to adopt open source methodologies without having to compromise future revenues from software sales by making their proprietary knowledge available under the GPL.

The GPL, and the MPL, however, were not the first licenses to govern free software. The most notable attempt other than Stallman's at building a free operating system was concentrated at Berkeley in the late '70s. Forking⁹ from early versions of AT&T's commercial versions of Unix, Berkeley's team released several versions of what came to be known as the 'Berkeley Software Distribution' or simply BSD. Unlike Stallman that was pursuing a social cause, Berkeley's BSD team members were not interested in changing the way people treat information, rather, they wanted to maintain the level of cooperation that was standard for academic research in general, and to continue to do their cutting edge research on Unix, which AT&T was trying to commercialize. The BSD License¹⁰ (BSDL) is, accordingly, a very short license, that can be considered as a codification of a venerable academic tradition of knowledge sharing: researchers can build off the work of others as long as proper credit is given to the original authors¹¹. Unlike the GPL, the BSDL does not require derivative work to be released under the same conditions, and for that reason we consider it as non-not 'viral'. The key requirement is to maintain the copyright notices of the original copyright holders and to give credit to original contributors. To this extent the BSDL serves well those developers that are mainly motivated by the increase in reputation, but its permissiveness in regards to derivative work is its Achilles heel: not requiring derivative work to be released under open source conditions renders projects in risk of 'hijacking'

⁹ 'Forking' is the process of branching a software development effort (into two or more development efforts), using an earlier code base.

¹⁰ The BSD license is available at < <http://www.freebsd.org/copyright/license.html> >

¹¹ N Bezroukov, BSD VS. GPL: A FRAMEWORK FOR THE SOCIAL ANALYSIS [Web] (2002 [cited 05/04 2004]); available from http://www.softpanorama.org/Copyright/License_classification/social_dynamics_of_BSD_and_GPL.shtml. gives a very detailed analysis of the diverse social factors that shape the GPL and the BSDL.

by interested parties that might be legally entitled to appropriate earlier works and use them in “closed” work. Many of the differences among the different OSL families results from different attitudes towards the handling of this appropriability problem (e.g. the inability of the innovator to appropriate the benefits of her creation in the absence of proper property rights), as will be discussed. To understand these, a short review of the symbiotic relationship between OSL and existing intellectual property protection mechanisms is in order.

Clearly, in order to grant a license it is necessary for a software developer to own underlying property rights – or gain permissions from the intellectual property owners to grant such a license. Traditionally an author of a piece of software has three legal mechanisms that can help her protect her work and maintain her property rights: copyrights, trade secrets, and patents. Trade secrets are perhaps the simplest traditional form of protection for software,¹² however, as much as technically software can easily enjoy trade secret protection, such protection is clearly a bad foundation for OSLs. Any license intended to maximize access to code refutes the whole purpose of maintaining information as a trade-secret. Given that at its heart an OSL regime offers unlimited access to source code (albeit if coupled with various restrictions in certain cases), in the context of OSL, trade secrets are clearly an inappropriate foundation for granting licensing rights.

Unlike trade secret protection which in the case of software aims to prevent competitors from accessing the secret source code, patents affords a much stronger protection: if granted they exclude would-be inventors from programming a specific software even if they would do it completely independently of earlier efforts. Patents potentially offer the strongest protection but

¹² Software is clearly included in the types of material that are entitled to trade secret protection under 18 USC 90. Moreover meeting the two conditions that the law requires (maintaining reasonable measures of secrecy, and deriving independent economic value from this secrecy) are very easy to fulfill in the case of software, even more than in other fields.

they are the most expensive to establish and to defend¹³ and in the significant challenges arise when considering patents and their use in the open source model¹⁴.

In a thorough enquiry of the economic considerations in the intellectual property protection of software Kenneth Dam discusses how copyrights are justified by their handling of the appropriability problem¹⁵ He finds that copyrights and patents as striking the optimal balance between limitations and incentives to innovation¹⁶. Granting creators the ability to control the distribution of their own work as well as derivative work is a means to ensure that said creators can indeed enjoy their creativity or, as the in the case of software, recoup the costs of research and development. As we know, copyright was originally granted to books and charts, and over the years was extended to include many more forms of cultural and scientific expression as these became available. Such extensions included protections to musical compositions and later moving images and, according to the Copyright Act of 1976 and the amendment of 1980¹⁷, to software. Whether including software under copyright protection is tenable over time or even a good idea to begin with is an ongoing heated debate¹⁸ but what all sides of this debate agree upon is that such protections should be balanced lest (a) the cost of maintaining them will rise to the point of

¹³ Especially in light of the requirement to establish novelty at the time of the application (given the size and fragmentation of the software industry), and the fact that patents are only meant to protect new ideas and not expressions of old ideas (a distinction which in software is hard to establish). How can one ascertain that her piece of software is useful and novel when most of the software in the world is closed? A search in the patent and trademark office files would not be very useful as it would not reveal those software projects that never applied for patent protection.

¹⁴ See for instance a detailed investigation of the issues in Sara Boettiger & Dan L. Burk, *Open Source Patenting*, J. INT'L BIOTECH. L. 221 (2004).

¹⁵ Dam, K.W. (1995) *Some economic considerations in the intellectual property protection of software*, JOURNAL OF LEGAL STUDIES, 24(2), p. 333

¹⁶ Dam, supra note 15 pp. 321-377

¹⁷ See GNU General Public License, supra note 4.

¹⁸ See Pamela Samuelson, *Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law to Computer Programs*, 70 MINN. L. REV. 471, 475-76, 507-11 (1985), and Pamela Samuelson, *CONTU Revisited: The Case against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L. J. 663, 741-53;

stifling innovation; or (b) such protections will imbalance innovation over time, preferring current over future innovation. As Dam explains:

...since [software] is an especially fast-moving technological field and since this new field is still fertile with innovations, the possibility that the pace of progress could be slowed by copyright protection cannot be ignored. Yet a failure to accord any protection would certainly slow progress, and even more immediately, since it would discourage software R&D expenditures. The objective of copyright law policy should therefore be to achieve an appropriate balance of innovation over time.¹⁹

However, Dam fails to notice, that the details of the licensing regime that relies on these property rights is just as important for innovation, given the intricate ways in which new software development efforts benefits from and builds on older endeavors. As we will argue, in order to achieve balance over time the licensing arrangement needs to balance the competing incentives of different groups of would-be innovators at different stages of development and innovation.

It is important to note that in essence OSLs have a symbiotic relation to copyrights. On one hand OSLs aim to abolish copyrights²⁰, but on the other hand they depend on copyrights in order to control the terms of use and redistribution of derivative work. Through the use of copyright law, OSLs seek a better balance for competing motivations for innovation, though founded on different assumptions as to the nature of such motivations. Boyle explains this relationship:

[N]either “free software,” nor most “open-source software” is in the public domain... [The GPL] rests on an intellectual property right, the copyrights held by the Free Software Foundation and other entities. The GPL says, in effect: Here is this copyrighted body of work; you may use it, add to it, modify it, or copy it— all of these uses are legal, but only if you comply with the terms of the GPL. Otherwise, your actions are infringements of the exclusive rights conferred by section 106 of the Copyright Act. ... In

¹⁹ Dam, *supra* note 16, p. 337

²⁰ See for example, Eben Moglen, *Anarchism Triumphant: Free Software and the Death of Copyright*, *FIRST MONDAY* 4, no. 8 (1999)..

legal terms, at least, the free software movement stands squarely on intellectual property.²¹

As noted earlier, the idea that 3rd parties may rely on underlying copyright protection to guarantee distribution reverses the traditional meaning of copyright (that is traditionally given to the original creator in order to prevent future re-appropriation of her work) and is, therefore, generally known as ‘copylefting’²². In copyleft, a copyright holder grants an irrevocable license to the recipient of a copy, generally permitting the free unlimited use, modification and redistribution of copies, with the condition that any derivative work will be redistributed under the same license terms. Copyleft applies copyright law to force derivative works to also become copylefted; since open source distributions include the source code, copyleft implies that derivative work redistributions must include the source code too, and reapply the conditions to its redistribution. For this reason copyleft licenses are considered viral. This viral effect, of course, is the heart of the free software paradigm. Over time a positive feedback loop is created and reinforced; as more and more work becomes copylefted the incentives for copylefting grow, driving even more software to be copylefted in turn. Not all OSLs are copyleft licenses. The clear example, as we have seen, is the BSD license family. Under non-copyleft licenses the derivative works can be released under commercial licenses or taken private.

In conclusion, it is clear that without this partial use of property rights, OSLs would find it almost impossible to motivate a large enough group of innovators that would be sufficient to power the wheels of innovation. For this same reason it is evident that putting software in the public domain will be inefficient. The key question that remains is how the details of different

²¹ Boyle, James. *The Second Enclosure Movement And The Construction Of The Public Domain*, LAW & CONTEMPORARY PROBLEMS VOL. 66 (Winter/Spring 2003) p. 33

²² According to Wikipedia, the free encyclopedia the term "copyleft" came from a message contained in Tiny BASIC, a free distributed version of Basic written by Dr. Wang in the late 1970s. The program listing contained the phrases "All Wrongs reserved" and "CopyLeft.". Available at < <http://en.wikipedia.org/wiki/Copyleft> >

OSLs use the traditional rights as they try to change the traditional delicate balance of power. We will be able to answer this after we investigate positive and negative effects of OSLs on innovation drawing important distinctions between different groups of innovators.

3 Innovation in Software

3.1 *Several Useful Distinctions: Developers Versus Users; Initial Developers Versus Later Developers*

The distinction between those who strictly develop software, and those who strictly use it does not always exist in the software field. Users are many times also developers who wish to customize previously developed software. Similarly, developers of new software are usually also users of previously developed software. Developers, who wish to license the software they develop, are many times not only licensors of software they develop, but also licensees of various pre-existing components used in their software. Given that users may also be developers, software terms of use may influence innovation in software.

We also find it necessary for the sake of the forthcoming analysis to make an important distinction between (i) developers that develop completely new software, and are the initial copyright owners, and can therefore control of the type of the license (if any) and its terms, under which they would make their software available. (we refer to them as *initial developers*, and to their work as *initial work*); and (ii) developers who create work that is based on *initial work*, or that incorporates components that are initial work (referred to as *later developers* and *later work*).

3.2 *The Sources of Innovation in software*

Innovation is complex—it may involve new products, new processes, new business models or improvements to any such products, processes and services. Innovation can sometimes revolutionize a sector in a few short years (e.g. Google's search engine) but more often it can only be measured with hindsight or over a long period of time (e.g. steam engines.) The benefits

of innovation may sometimes give the innovator a long-lasting advantage in the marketplace (e.g. Eastman-Kodak) but often they accrue to many and not only to the innovating party itself assisting, for example, partners or even rivals to become more competitive. Innovation can be scrutinized from different perspectives as an economic, social, or technological phenomenon, point of view which would all indicate one important aspect: there is a great diversity in the sources of innovation and the motivation for it. This section highlights the fact that licensing schemes, which typically involve conditions and restrictions that govern the relationship between earlier developers and later developers or, more commonly, the relationship between developers and users, must take this diversity in the sources and motivations of innovation into account in order to maximize innovation.

First, we note that it is still widely (and wrongfully) believed that innovation derives primarily from the activities of manufacturers. In software this translates to the view that most innovative software will come from firms that develop software as their main business and their battles with one another. To see how prevalent this view is, it is enough to point to the amount of resources governments around the world (and particularly the US Department of Justice) spent on regulating Microsoft's monopolistic behavior on the grounds that they block competition and stifle innovation²³. But as recent evidence suggests, the central role attributed to manufacturers might be all too exaggerated; in software, as in many other fields, innovation can actually be equally induced by users and suppliers. This diversity is what Eric von Hippel calls the variance in the *sources of innovation*²⁴. In a careful analysis of a detailed 12 year field study of four

²³ See United States of America vs. Microsoft Corporation, (1988) Civil action no. 98-1232 (antitrust), Complaint, District Court for the District Of Columbia. §11,36-37. Available online at < <http://www.usdoj.gov/atr/cases/f1700/1763.htm> >. The complaint revolves around the "browser wars" and it portrays Internet users as passive receptors of technology while innovation is described as a processes that is encouraged by the competition of manufacturers, in this case Microsoft and Netscape.

²⁴ Von Hippel Eric (1989) THE SOURCES OF INNOVATION. Oxford: Oxford University Press.

scientific fields von Hippel argues that the sources of innovation are diverse and that innovation is just as likely to stem from pioneering users or innovative suppliers as from manufacturers. Von Hippel defines functional relationships among these three classes (suppliers, manufacturers, users) and argues that innovation is likely to occur by a party that stands to gain direct short-term economic value from such innovation (“collecting economic rents”) in an environment in which it cannot easily switch its functional role, and in which it cannot easily benefit from licensing the invention to others. These conditions are generally prevalent in software, so we can expect that in software users will play a significant role as innovators, and indeed the open source phenomenon is a case in point. We must then investigate the ways by which software innovators will collect such economic rents, specifically if they don’t rely on licensing fees, as is clearly the case in open source innovation. In point of fact, the ‘motivation question’ was the focus of a significant portion of both theoretical and empirical first generation open source related research²⁵. However, after several years of research, there is no agreement in the literature on what the primary motivation factors for open source participation are. Explanations vary from career management concerns and market signaling incentives (Lerner and Tirole), through gift culture reciprocities (Raymond) and a hacker ethic (Himanen) to personal profits induced by the non-rival nature of software (Weber), and yet what clearly comes out of all these accounts is that users play a crucial role in software innovation regardless of their motivation. In another study, von Hippel and von Krogh discuss open source innovation in light of the two traditional models of innovation, that of private investment and that of collective action:

²⁵ See Lerner, J. and J. Tirole (2000), *The Simple Economics of Open Source* < <http://ssrn.com/abstract=224008> > initially published as *Some Simple Economics of Open Source*, 52 JOURNAL OF INDUSTRIAL ECONOMICS (June 2002) 197-234. See also Lakhani, K. and E. von Hippel, *How Open Source Software Works: 'Free' User-To-User Assistance* Working Paper 4117, MIT Sloan School of Management, Cambridge, MA (2000). and Lakhani, K., R. Wolf., *Does free software mean free labor? Characteristics of participants in open source communities*, BOSTON CONSULTING GROUP SURVEY REPORT, BOSTON, MA. (Available at < www.osdn.com/bcg/ > See also Weber, S. (2004). *THE SUCCESS OF OPEN SOURCE* Cambridge, Mass., Harvard University Press. Ch. 5-6. See also Raymond, E. S. (1999). *THE CATHEDRAL AND THE BAZAAR : MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY*. Sebastopol, CA, O'Reilly. See also Himanen, P. (2001). *THE HACKER ETHIC, AND THE SPIRIT OF THE INFORMATION AGE*. New York, Random House.

The “private investment” model assumes returns to the innovator result from private goods and efficient regimes of intellectual property protection. The “collective action” model assumes that under conditions of market failure, innovators collaborate in order to produce a public good. The phenomenon of open source software development shows that users program to solve their own as well as shared technical problems, and freely reveal their innovations without appropriating private returns from selling the software.²⁶

The authors argue that in effect open source software development is an example of a hybrid model of innovation, what they call the “private-collective” model, which combines the private investment and the collective action models and improves upon them: “participants in open source software projects use their own resources to privately invest in creating novel software code. In principle, these innovators could then claim proprietary rights over their code, but instead they choose to freely reveal it as a public good. Clearly, the net result of this behavior appears to offer society the best of both worlds—new knowledge is created by private funding and then offered freely to all.”²⁷ The argument further goes to list the benefits that accrue to open source participants, including the joy of learning, elevated reputation, and access to bug reports and fixes that would otherwise be costly to obtain. These benefits come in lieu of the traditional benefits expected according to the private investment model, namely, the prospect for economic profits from licensing fees protected by a strong intellectual property regime that enables such for-profit licensing. What is important to remark upon here beyond von Hippel and Von Krogh’s point, however, is that as much as the models differ in the use of software licensing, just like in the private investment model so in the private-collective model too, licensing is the cornerstone that holds the entire private benefit edifice in place. It is also important to note that as much as users in general, and open source users/developers particularly, play significant roles as innovators, most software today, at least quantitatively, is still licensed under non open source

²⁶ Von Hippel E. and G. von Krogh, (2003) *Open Source Software and the ‘Private-Collective’ Innovation Model: Issues for Organization Science*, ORGANIZATION SCIENCE Vol. 14, No. 2. p. 217.

²⁷ Von Hippel and von Krogh Supra note 26, p. 213

licenses. Consequently, as we argue in detail in the following sections, we can expect that the details of the open source licensing mechanism that governs the levels of private benefit and collective action in the system, , will be influential in directly affecting the degree of innovation. Moreover, it is exactly in the connection points between the open systems and the proprietary ones that these details matter as they can affect the delicate balance that exists between the different motivations.

4 Software Licenses and Innovation

4.1 *The effect that licenses and the availability of licensing schemes may have on innovation*

The availability of various licensing schemes, may affect both the motivation of initial developers (as we have previously defined them) to make their work available to others (at cost, or in exchange for the fulfillment of other conditions), as well as the ability and motivation of later developers to innovate, to the extent that they are required to license existing software components that are necessary for their later innovation.

Although initial developers may also be later developers with respect to portions of their work (if they are required to license certain components in order to create a whole work), for the sake of distinguishing between different sets of motivations and for analytical cause we shall treat them as exclusive copyright owners, who only decide whether and under what terms to license the software they develop.

Software licenses provide individuals who are not owners of the property rights of the underlying software, with various types of rights with respect to the software. These rights may include any of the following: (i) the right to use the software (sometimes for only limited permitted purposes); (ii) the right to modify or customize the software, including the right to access the software source-code; (iii) the right to create any type of derivative work based on the

software and to link (dynamically/ statically) to the software; and (iv) any of the following rights to market, resell or sublicense the software (or a combination of such rights). Each of these rights may be granted on an exclusive or a non-exclusive basis. They may be granted on an individual basis, or as a public license. Licenses may also include various types of warranties as to the performance of the software and maintenance of the software, and may be limited in time or alternatively be perpetual.

Typically, rights that are not granted explicitly vest with the owner of the software. Licenses also include conditions to granting any of these rights. While in proprietary closed-source software licenses these conditions relate primarily to compensation and fees paid by the licensee, as well as sometimes to the performance of other contractual obligations by the licensee, the rights typically granted in OSLs are usually contingent upon conditions related to maintaining the open nature of the software, as well as providing recognition to its developers.

It is evident that software licenses can govern the permitted uses of both executable software (“binaries”) and its underlying source-code (“source”). To the extent that new innovations require the use or incorporation of certain existing software components, which are governed by a license, the terms of such license may greatly impact the likelihood and speed of such later development. In general, materials for new innovations could either be widely available for use (e.g. if they are part of the public domain) or they may be unavailable for use (e.g. if they are proprietary, and their owners do not wish to grant permission to use them.) They may also be proprietary but available under a variety of licenses. To the extent software components, licensed by initial developers are necessary or essential for later development, the terms and conditions under which they are licensed may greatly impact the likelihood of later innovation based on them.

A key determinant of the effect that the terms and conditions of a license may have on subsequent innovation is the type of motivation for such later innovation. For example, where innovation is motivated by financial gain, any elements in a license, which could impact such gain (for better or worse) are most relevant. Alternatively, where innovation is motivated by fame, need for recognition or other modes of a reputation economy, acknowledgments and publicity are the most relevant factors. Where innovation is motivated by the mere desire to develop, learn, and innovate maybe no terms are required, or terms that lead to the maximum later innovation and development are the appropriate ones.

Another important determinant is the motivation for developing the initial work. If such work is motivated by the opportunity of financial gain, the ability to license the initial work in a closed and proprietary fashion will be important. If the motivation of the initial developer is purely innovation related, yet such developer would like to prevent exploitation by opportunism of later developers, the availability of alternative licensing schemes, as well as the ability to control property rights and prevent opportunism via the use of licensing schemes could also serve as an important stimulant for such initial innovation.

In the following section, we shall examine how the design of OSLs may influence the likelihood that initial developers will resort to them, as well as the likelihood that later developers will incorporate source code released under an OSL in their later development.

5 OSLs and innovation

OSLs' impact on innovation will be examined as a function of their attractiveness to initial developers (and consequently their impact on the amount of code made available and released under such licenses), as well as a function of the likelihood that later developers will develop later work based on the code released under an open source license, and the likelihood that their work will facilitate further development. The questions we examine are, to what extent

OSLs facilitate making a greater amount of software available for later development, and to what extent such software and code are indeed useful for later developers, given the terms and conditions imposed by certain OSLs. It is not only sufficient that initial developers will be motivated to grant access to the code underlying their software. It is also essential that such code indeed serve as a foundation for additional development, and that there be potentially granted access to the code underlying such development as well.

While there are many flavors of OSLs, which may significantly differ from each other, the open source definition requires that these licenses meet certain minimum terms. These mainly include the stipulations that (i) there are no restrictions are imposed on the redistribution of the software; (ii) that the distribution must include source code, and must allow distribution in source code as well as compiled form; (iii) that the license must allow modifications and derived works²⁸; and (iv) that the integrity of the author's source-code is maintained. While on the one hand OSLs grant unlimited access rights to source code, thereby making more resources available for future innovation, on the other hand, some of these licenses impose restrictions on the rights to distribute and license work based on code released under such terms. In certain cases (for example in the case of using the GPL) the restrictions are such that developers, who wish to financially capitalize (directly) from their development work (derived from the code covered by such OSLs), may not be able to do so. In other cases (such as the BSD license) there are no material restrictions on the rights to use, distribute and license modified and derivative work based on the original code. Depending on their motivation to develop software initial developers may find certain OSLs more attractive than others. For example, while certain initial developers may resist using the BSD, in light of potential appropriation of their work by later developers,

²⁸ and must allow them to be distributed under the same terms as the license of the original software

others may not mind that their code be included in others' software projects later on be them commercial in nature or not. As we later conclude, while each of these licenses addresses certain motivations of developers to grant access to their source code and comply with other requirements of the open source definition, only provisions of any of these licenses can actually offer better grounding for maximum innovation.

In a recent extensive empirical survey performed by Josh Lerner and Jean Tirole²⁹, 40,000 open source projects were analyzed in an attempt to explore the various considerations that figure into the licensor's decision of how restrictive a license to employ. Their work highlights the complex set of motivations that may drive the choice of a license. They distinguish between three classes of licenses: unrestrictive (for example, the BSD license), restrictive (e.g., LGPL), and highly restrictive (GPL). Their research suggests that permissive licenses will be more common in cases where projects have strong appeal to the community of open source contributors, and restrictive ones commonplace when the appeal is more fragile. They find that projects geared toward end-users tend to have restrictive licenses, while those oriented toward developers are less likely to do so. While this research tries to match between license choice by initial developers and their motivation to develop, it does not examine whether current licenses are indeed designed to maximize their appeal to initial developers, or the extent to which OSL choice by initial developers influences the ability of later developers of various types to incorporate initial work in their later development. Preliminary results of an empirical survey directed at the latter questions conducted by Tsur and David show that developers motivated by

²⁹ Josh Lerner and Jean Tirole, *The Scope of Open Source Licensing*, NBER WORKING PAPER 9363

financial gain, are reluctant to include software components governed by OSLs, especially if those are as restrictive as the GPL³⁰.

5.1 *What type of a license is an open source license?*

We note that OSLs are non-exclusive public licenses (a license offered by an owner of intellectual property to the public on a non exclusive basis). In general, OSLs address the rights to access the underlying source-code as well as the rights to use, exploit, distribute, modify, customize and create derivative work based on the underlying software and its underlying source-code. Whereas typically these licenses grant unlimited rights to use and exploit the software, as well as unlimited rights to access the source-code of such software, they also include varying conditions that govern the distribution modification and creation of derivative work based on the software (and its underlying source code). In most cases they also disclaim any warranty on behalf of the licensor.

As mentioned earlier, though OSLs are treated as a “category” of licenses, there are many differences among the various types of licenses. Meeting the requirements of the open source definition is not a strong enough ground to unification. Some of the differences among the licenses affect the motivation of both initial copyright owners (to release their code under an OSL), as well as later developers who wish to use and modify in their later innovations code previously released under one or more OSL. The details of these conditions depend on the specific OSL chosen. Typically, we find that the main difference among the many flavors of OSLs has to do with their treatment of modifications and derivative work, and the distinctions that some of them make among commercial and non commercial use.

³⁰ Michal Tsur and Shay David, *Open source licenses and innovation*, Working paper, Information Society Project – Yale Law School.

As suggested in the previous section the nature of the rights to access source code, to modify it, as well as rights to distribute and sublicense modified and derivative work based on the software, may all affect innovation based on the licensed software. The ability of later developers, who may want to access and use source-code of existing software, to utilize previous work covered by an OSL for their later development, and the extent to which they will indeed do so depends on the exact limitations and conditions set with respect to the modification and future licensing and use of modified and derivative work. We should note that there are some cases in which access rights to source code are granted under commercial “proprietary” licenses too. Such licensing arrangements, however, do not usually grant such rights to the whole developer community, but rather to individuals, on a case by case basis³¹. Usually the licensors require significant compensation (in a variety of forms) for disclosing the code, or burden the licensee with the demand to prove that the licensor is incapable of maintaining the software, as in the case of source-code escrow agreements. Moreover, the differences among licenses (open or proprietary) with respect to the conditions of source code access affect the decision of the initial developers as to which of them to use when releasing the software. Particularly, while certain OSLs may have greater appeal to initial developers, such OSLs may not necessarily appeal to later developers who may have concerns about restrictions imposed by certain OSLs. Moreover, it is also possible that initial developers do not always make informed choices of OSLs, and may select by default the de-facto OSL standard, the GPL, without seriously considering later impact it may have. Initial results of an empirical study conducted by Tsur and David, support this possibility³².

³¹ Under this category we find, among others, Microsoft’s “shared source” licensing arrangements. See <http://www.microsoft.com/resources/sharedsource/Licensing/default.aspx>

³² See Tsur and David, *Supra* note30.

5.2 OSLs and Current vs. Future Innovation

By distinguishing between the attitude of initial developers (which we treat as the exclusive copyright owners) and later developers towards OSL terms, we are better able to analyze the incentives of developers to not only release code under an OSL, but also to make use of code previously released under an OSL. It is thus evident that OSLs may affect the availability of resources for innovation as well as motivation to innovate of both initial and later developers.

Clearly, the existence of OSLs (despite their confusing variety of forms) increases the variety of licensing schemes under which developers may opt to release their software. To this extent, OSLs represent an additional form of licensing and, therefore, OSLs can only promote and increase innovation to the extent that the existence of the option to release software code under an OSL addresses the initial developer's motivation. The later developers, however, have a different set of considerations. If later developers wish to create work based on existing software, depending on the copyright owners' (i.e. the initial developers') approach, they may be required to either: (i) acquire the rights to create work based on the initial work under a variety of forms of commercial licenses; (ii) may be prevented from using such code if the copyright owner does not wish to license the code; (iii) may be free altogether to use the code if the code is released to the public domain; or (iv) may be able to use the code subject to conditions and restrictions imposed by an OSL.

To the extent that later developers are motivated by the financial gain they may derive from their work, if initial work is released under certain types of OSLs which restrict the ability of such developers to derive such gain from their work, such initial work could not serve them as a resource for their inventions³³. There is no fee that the later developer can pay in order to be

³³ In essence, some of the OSLs require that if a developer chooses to use the work covered by an OSL as a basis for her innovation, she will not be able to exclude others from her work.

able to distribute its derived work otherwise. Provisions in OSLs which are contrary to such motivation of later developers, may have a negative impact on innovation. If however later developers are otherwise motivated (for instance by recognition, the desire to belong to a certain social network or group, or to otherwise increase their reputation) the imposition of restrictions and conditions that prevent monetizing on the development would not negatively impact innovation in such cases. It is thus easy to see that the impact that a certain OSL has on innovation is derived exactly from how it addresses both initial developers' and later developers' motivation to develop.

5.3 Aspects Of OSLs That Positively Impact Innovation

Certain provisions in OSLs may have a positive impact on innovation, by either increasing the likelihood that initial developers will use an OSL at the outset (and hence increasing the “public” repository of source code, available for later development, or for other uses, such as education), or distribute their software (whereas they may have kept it to themselves without the affordances of the OSL option), as well as by increasing the likelihood that indeed later developers will be able (given their varying motivations) to build on previous work released under an OSL, hence accelerating the speed of their development. As we discuss, whereas certain aspects in OSLs address the motivations of initial developers, other aspects address those of later developers (or both).

For a sector to thrive over time a constant supply of fresh minds is necessary and accordingly the indoctrination of younger generations is part of any intellectual discipline, software notwithstanding. Like in many other fields, in software too the best type of education calls for hands-on trial and error and continuous practice which involves tinkering with existing software, and improving upon it. Particularly, learning how a piece of software works without access to the source code is exceedingly hard, as Andrew Tannenbaum a computer scientist and

educator from the Free University in Amsterdam, and the creator of the Minix operating system (which may be considered as the forerunner of Linux) knows very well. During the 1970s Tannenbaum appreciated teaching Unix which was an open system at the time. Tannebaum was very disappointed then when in 1979 AT&T, the owner of Unix, shipped Version 7, which was the first closed version of Unix in which the source code was no longer available to anyone, including his students. He thought of ways to bypass the problem but reached the conclusion that: “[the only solution] was to write an operating system on my own that would be system-call compatible with Unix...but which was my own code, not using any AT&T code at all.”³⁴ Minix became a very stable open operating system, and in fact the system of choice for operating systems’ education in many computer science centers around the world, including the university of Helsinki where several years later Linus Torvalds would outgrow Minix and invent Linux. The point is that access to the code is an essential ingredient of software education. OSLs, by assuring access to the source code, promote the level of education, which soon thereafter translates to innovative software. Historically, students in varying situations and within different segments of the software market contributed to major software innovations even before completing their formal education-- or sometimes instead of completing it³⁵. In essence, OSLs allow students to become ‘later’ developers instead of forcing them to remain ‘initial’ developers.

The effect of source code access on education continues well beyond academic centers. Within industry too, and throughout a programmer’s career, exposure to the software of others has a long lasting effect on one’s dexterity as a programmer and gaining access to the source code is arguably significantly more useful than just seeing the software execute. Naturally, well trained

³⁴ Moody, Glyn. (2001) REBEL CODE : THE INSIDE STORY OF LINUX AND THE OPEN SOURCE REVOLUTION. Cambridge, Mass.: Perseus Pub. P. 33

³⁵ e.g. Bill Gates and BASIC, Larry Page and Sergey Brin and Google’s search technology.

programmers are better positioned to pursue innovation, but in addition it is important to realize that OSLs motivate programmers to bring to an end unfinished work left by others and/or to improve on their work. Thus, if during the learning process a student finds a bug in the code that she fixes for her own needs, or better yet, improves the software in one or more ways, provisions in OSLs may encourage or require (depending on the license) that this fix be introduced to the central repository. Thus innovation is built into the learning process.

Another advantage facilitated by OSLs is that of using existing software as scaffolding when building a new software project. A common practice in large software projects is the use of intermediate software modules that perform approximate functionality and can hold the project together until there is time to finalize each module (e.g. when writing a program that is intended to manipulate images, the module for saving and retrieving files of an earlier software can be used until specific functions can be written to support all the peculiarities of the new software.) In a closed environment a firm would have a hard time finding such appropriate modules. In an OSL controlled environment, finding such appropriate components is very easy including potential use of modules from competing products. Evidently, such a practice accelerates innovation. Again, OSLs address the difficulties of ‘later’ developers that otherwise might need to replicate work that was already performed by the initial developers.

And from the last point we get to perhaps the most significant positive effect that OSLs may have on innovation. A common practice when designing and coding large and complex software applications is the use of divide-and-conquer techniques. The software is divided into modules, which perform simple functionalities, and these are conquered by the programmers in turn. Smart modeling of the software is the basis of efficient code re-use and re-factoring (e.g. if the functions of saving and retrieving files are abstracted in the proper manner then a “File-Handling” module could potentially be used in different types of applications regardless of the

contents the files it handles.) In a closed environment, there is a huge amount of replication of efforts. The same modules are being written over and over again all across the industry because the only modules that a firm's programmers can re-use are modules that they themselves wrote at an earlier stage, or modules that they buy from third parties³⁶. Within firms there is often internal resistance to the use of third party components either because the functionality offered by components available in the limited marketplace that exists for such components does not match the exact functionality needed, because of architectural mismatches, or because they such components are expensive (either absolutely, or psychologically, since buying external components is an out-of-pocket expense compared to the amortized costs of an ongoing R&D effort.) The result, of course, is that significant portions of software development efforts in the industry are aimed at re-inventing wheels rather than innovating³⁷. OSLs facilitate a simple solution to this problem. By creating central repositories for projects, OSLs ensure that a large base for code re-use is created. Some of this code can be in the form of libraries or other well-formed modules that can be used off the shelf, for free or for a small fee, but not less importantly, much of the code can be used in even smaller units. A well trained programmer can easily stitch files and functions appropriated from other projects in a very short time even if they were not released as stand alone libraries so long as their functionality is well defined and well documented. The amount of code that is thus available under OSLs is tremendous compared to the amount of components available in commercial component marketplaces. This is easily explainable if we remember the earlier point regarding the important role users play in software

³⁶ Such software is generally referred to as: Commercial Off The Shelf – COTS

³⁷ As we know, such re-invention of wheels is economically inefficient. Potter, for one, argues that “coupling copyright with restrictive licensing terms leads to economically inefficient results” because programmers are forced to spend resources on problems that are already solved by others. See Shawn W. Potter, *Opening Up to Open Source*, 6 RICH. J.L. & TECH. 24 (Spring 2000). For a general description of the implications of COTS see Oberndorf, T., COTS AND OPEN SYSTEMS - AN OVERVIEW, 1997, available at < <http://www.sei.cmu.edu/str/descriptions/cots.html#ndi> > Also see M. Morisio, C. Seaman, A. Parra, V. Basili, S. Condon, and S. Kraft, (2000) *Investigating and Improving a COTS-Based Software Development Process*, PROCEEDINGS OF THE 22ND INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE 2000), Limerick, Ireland, June 2000.

innovation. Many of the modules, classes, and functions available under OSLs are written by students, programmers in academic centers, or internal programming staff in large firms whose main business is not software, all of which would not make an effort in a closed environment to release for-profit components, simply because it is not their main business. The result of all this is clearly that the programmers can focus their efforts, resources, money, and energy on creating innovative products rather than re-inventing old ones. The aggregated positive effect on innovation and its dependency on OSLs is evident.

Another positive effect on both initial and later developers is that the open model allows for better scrutiny and criticism at different project stages. This ability to scrutinize other people's work in near-real-time results in one of two outcomes: either projects are cancelled early on³⁸, or the projects that are continued maintain high quality of coding throughout their development cycle. Collective action is the key concept here, where the joint goal is to create safe and reliable software. Such action works at two levels: (i) what is escapable from the eyes of one software designer is inescapable from the eyes of another; people can find bugs just by looking at the source code that other people wrote; and (ii) by releasing many versions of the software as early as possible testing is done in the real-world and not in simulated environments—this guaranties that the maximum amount of possible code paths be executed and, at the same time, it guarantees that as many external (environmental) variables are internalized since what constitutes a far-fetched operation scenario for one person is the everyday state of affairs for another. Of course,

³⁸ According to an analysis of data available from SourceForge.net (the largest site for the management of open source projects) the distribution of projects according to stages of development in August 2004 suggests that only 41% of the projects are in working status (e.g. Beta stage or more mature stages). Raw data available at < http://sourceforge.net/softwaremap/trove_list.php >. This information is often read against the open source model, suggesting that many of the projects never mature. In contrast, we read this data as evidence of an efficient resource allocation mechanism; projects that cannot solicit interest from the development community are left in early stages, without many resources being wasted as would happen in a closed environment. This often happens as developers join similar but more advanced projects.

all these advantages hinge directly on the mandate that any and all changes to the source code be released to the central repository as specified by the OSL.

One last advantage can be observed in terms of increased efficiency of resource allocation. According to Benkler, the open source model, which is part of what he calls the peer production model, facilitates better resource selection and assignment within each project. He writes:

Peer production has an advantage over firms and markets because it allows larger groups of individuals to scour larger groups of resources in search of materials, projects, collaborations, and combinations than is possible for firms or individuals who function in markets. Transaction costs associated with property and contract limit the access of people to each other, to resources, and to projects when production is organized on a market or firm model, but not when it is organized on a peer production model³⁹.

In other words, OSLs facilitate a reduction in transaction costs that would otherwise grow if would-be collaborators had to negotiate mutual work on specific source code pieces in a closed-source environment. This reduction in transaction costs is especially important where human creativity is often the scarce resource that limits innovation. OSLs, thus, have a positive effect on both initial and later developers.

5.4 Potential Negative Effects of Certain OSL Provisions on Innovation

As discussed in previous sections, the various OSLs differ significantly in the conditions that they impose on later developers to using initial work. The more an OSL restricts the freedom of later developers, and imposes conditions to using the code of initial work, the less likely are such developers, depending on their motivation, to use existing code that is available under a restrictive OSL. If conditions imposed by an OSL are such that later developers in essence are prevented (due to their objectives) from utilizing the code, many of the positive effects (on

³⁹ Benkler, Y., *Coase's penguin, or, Linux and The Nature of the Firm*, YALE LAW JOURNAL Vol 112

innovation) associated with making the source code available might not exist. Moreover, to the extent that such code could have served as a foundation for many other projects, innovation may be stifled if later developers will refrain from building on such earlier work due to certain OSL provisions. This section shall focus in its analysis on the restrictive GPL family, whose effect is most significant due to its overwhelming widespread use, and on the BSD family, which represents the least restrictive type of licenses.

Interestingly, the GPL is the most restrictive license, in terms of the freedom of later developers to distribute and license derivative work based on such work. Moreover, given that many developers view and treat the GPL as a representative OSL, their attitude towards code released under other OSLs may be influenced by the content of the GPL, even if other OSLs may well be less restrictive. The key negative effect of the GPL is stirred by fear of the GPL viral clauses. The GPL allows users to use, copy, distribute and modify GPL code, provided that the user licenses any derivative work under a GPL license as well (hence, this condition is referred to as a “viral clause”). Users must also agree to not establish proprietary rights in the software, to provide access to the underlying source code, to include in the software notice that the code is subject to the GPL; and to accept the GPLed software with no warranties. Whether or not later developers will create “work based on” initial work released under the GPL, depends to a great extent on whether they will indeed be required to release their later work under the GPL as well (as this latter requirement is the only significant one in terms of substantial limitations that it imposes on the later developer). If they are indeed required to do so, and if their prime motivation for development is financial, consequently the effect of the GPL will be to limit such innovation.

There is however certain ambiguity as to when later work constitutes “*work based on...the initial work*”. Theoretically, later developers are required to subject their code to the terms and conditions of the GPL only to the extent that their later work is considered “Derivative

Work”. However it is not absolutely clear what constitutes a “Derivative Work”⁴⁰. A Later Work is generally considered a derivative of an Earlier Work if it contains a substantial amount of material from a preexisting work. It is not clear, however, what sort of technical activities fulfill this condition. For example, it is not clear whether the essential process of linking, which brings together different software modules (e.g. software libraries) that are designed to work together and merges them into one executable program makes a work derivative or not. The answer to this question depends on the type of linking involved and the interpretation of the GPL. Under static linking, different software modules are more tightly-bound than under dynamic linking. The GPL does not clarify precisely which type of linking is permitted. Clearly if both dynamically linked code, as well as statically linked code are “derivative” work for the sake of the GPL, developers of code which requires such linking, may not be able to use the GPLed code, if they wish to capitalize on their work. As long as the answer to this question is unclear, risk-averse developers will hesitate prior to assuming the risk of having to subject their code to the GPL (assuming their motivation is financial).

Though, as noted earlier, significant innovation in software is being driven by users, we cannot ignore the large portion of software production that is still performed for the sake of profit, and is often innovative in its own right. The limited “monopoly rights” granted by the intellectual property regime are probably an important, if not crucial, incentive for investment in such traditional software innovation. To the extent that earlier work released under GPL is necessary as a basis for the development of a later work by developers motivated by profit, for whom the

⁴⁰ It should be noted in this respect, that the GPL itself with its definitions contributes to yet further ambiguity. See Matt Asay, *A Funny Thing Happened on the Way to the Market: Linux, the General Public License, and a New Model for Software Innovation* (Unpublished, 2002, available at <http://www.linuxdevices.com/files/misc/asay-paper.pdf>), who argues that this ambiguity was actually resolved in the marketplace, and poses not risk to innovation. He argues that a convention has developed under which loosely-coupled projects (e.g. those who only make function calls and are not compiled together with the original) are not considered derivative work. We disagree with Asay and believe that although such practicalities are being negotiated many miss-clarities and disagreements remain even within the small group of key open source figures. These ambiguities, we argue, would be better solved in the terms of the OSL itself and not left for negotiation.

intellectual property rights in the later work are essential, the later work might not be developed, or its development may be delayed. In such a case, the fact that the GPL fails to address the motivation of developers driven by financial gain, may reduce or slow down innovation. Moreover, many initial developers are not legal experts. Given that the GPL is treated by many as the prominent OSL license, it often becomes the default choice of initial developers, whose work serves as a basis for later developers. From the point of view of later developers, including corporations, who consider utilizing code that was initially released under an OSL, the risk related to the obligation to subject any derivative work to the GPL prohibits their use of GPLed code. Moreover, the uncertainties with respect to the impact of the GPL, also negatively impact the attitude towards work covered by other OSLs. Hence, potentially useful code, which is released under the GPL, or even under a different license (which may mistakenly be perceived as similar to the GPL in its viral effect), might not be used in later development depending on the motivation of the later developers.

One additional negative effect of a too-strong OSL might come as a surprise. If the OSL does not strike the correct balance between initial and later developers' motivations it might create the parallel of a software monopoly. This situation might arise, as Steven Weber points out if the conditions of the OSL serve to greatly increase a first-movers advantage⁴¹. In such a situation a developer or firm that has control of a project might use the viral clauses of the OSL to stifle innovation by limiting competitor's ability to offer innovative features in the marketplace. Because any additions or improvements to the software must be released together with their source code, later developers cannot hope to make any substantial financial gains by improving

⁴¹ Weber, S. (2004). THE SUCCESS OF OPEN SOURCE. CAMBRIDGE, MASS., Harvard University Press. pp 221-2.

upon an existing product. This situation results in vendor lock-in in the same way that such lock-ins develop in the proprietary software arena.

The BSD type license is a much more permissive license, and does not impose restrictions on later developers, who are therefore likely to make use of initial work, whose code is released under this license. While this license may have a positive impact on innovation by later developers, and indeed is well suited for varying motivations of later developers, it fails to address the varying motivations of initial developers, some of which may not want later developers to be able to capitalize on their work, and include it in closed software. Hence, it fails to provide a strong enough incentive for initial developers to release the code of their initial work.

6 Drafting OSLs That Maximize Innovation

This section suggests that introducing certain changes to existing OSLs, or to the Open Source Definition, may cause more initial developers to release software under the revised OSLs, and importantly, may cause more diverse later developers, including developers who are motivated by financial gain, to make use of code released under such revised OSLs. Increasing the size of the repository of code available for later development is not sufficient in itself, if later developers may be prevented from making use of such code because of the restrictions, or conditions imposed by the OSLs. Similarly, if revised OSLs only consider later developers' motivations, there may be fewer initial developers willing to release code under OSLs. It is necessary to perform a delicate balance of the motivations of both types of developers.

6.1 *Is Innovation A Worthy Objective For OSLs (Why Is It Important To Craft OSLs With Innovation In Mind)*

The key justification for granting monopoly rights to inventors and innovators is that such rights would further innovation and, overall, benefit society. Clearly, licensing schemes that try to address innovators incentives in order to maximize innovation are desirable. Though OSLs

had not been initially crafted with this objective in mind, but rather with the objective of revolutionizing the nature of collaborative software projects, and facilitating the exchange of information without the fear that such information (once released) be appropriated, the goal of achieving maximum innovation is definitely not in conflict with such initial objectives.

Moreover, our working assumption, which is backed up by observing the millions of developers that are participating in open source projects, is that there is a growing group of developers that are interested in maximizing access to their code, and are in need for fitting software licenses to accompany their projects without jeopardizing their other motivations (e.g. fame or fortune.) For such developers--whether initial or later ones--OSLs are a primary tool in the promotion of their own interests, though most likely they all share a deep commitment to developing innovating software. By showing their interest in the open source models these developers demonstrate their interest in promoting innovation, otherwise they could have resorted to standard commercial software licensing arrangements. Indeed open source developers would support changes that both act to better address their motivation to develop, as well as promote innovation.

6.2 Current OSLs Are Not Consistent In Furthering the Goal Of Maximizing Innovation

As described in previous sections there are many flavors of OSLs, each of which addresses different motivations of both Initial as well as later developers. To be sure, the ability to release proprietary code under an OSL only increases the options available to initial developers to distribute their code, and as such OSLs always contribute to innovation. To what extent, if at all, does the possibility to release code under any of the existing OSLs motivate initial developers? Might initial developers refrain from developing if OSLs did not exist? It is quite difficult to answer these questions without an in depth empirical survey, however, we can reduce these questions and ask whether developers would refrain from sharing their work with additional

developers absent the OSLs. The answer to this last question is that initial developers would probably share their work to a lesser extent, if they were concerned about opportunistic behavior of later developers, or if they wanted to control the extent to which later developers could build on and gain from their initial work.

Currently the two primary distinct types of licenses – either completely assure the initial developers that their work would never be appropriated (the GPL license), or do not provide them with such assurance (the BSD license– pursuant to which the initial work can be included in proprietary code.) Initial developers may not resist altogether to the inclusion of their work in proprietary code – if for example they knew that they would receive recognition for such inclusion, or if, for example, they knew that the proprietary nature is limited in time.

Later developers are affected by the licensing scheme selected by the initial developers in two main areas. First it will affect the extent to which they have access to existing inventions, as well as the manner of permitted access. Second, the license under which initial work is distributed, may affect the manner in which they are permitted to distribute their later work, and the terms of such distribution. While the BSD License is not restrictive, and therefore would satisfy most types of later developers, restrictive licenses such as the GPL (as discussed in the previous section) do not address the needs of financially motivated developers.

6.3 *Suggestions for Improvement*

The key motivation for improving OSLs is to accommodate the competing needs of for-profit developers that have hitherto chose to release their software under the terms of proprietary licenses with those of developers (volunteers or companies) whose business model, or motivation supports outright open source models. The key metric we should look at in this regard is the size of the common repository. It is possible that certain amendments to the licenses would enable entities that have hitherto stayed away from open source models to partake in them, thus not only

increasing the size of the repository (and reinforcing the positive feedback loop that such a repository creates), but also assuring that initial work released under such OSLs could be used as a foundation, not only for later work of a small open source community, but also as a foundation for later work of profit motivated developers.

Improved OSLs need to address not only the later developers' incentives, but also and importantly the initial developers' incentives. If initial developers will not resort to these licenses in order to release their code, such OSLs would not be instrumental to innovation. Initial innovators may belong to several main types: (i) Those who wish to release their work commercially; (ii) those who wish to release their work commercially, but would be happy to contribute the work for further research and development, if they are assured that no one appropriates their work; (iii) those who initially released their work commercially but think that they could better benefit if their work served as a basis for further development; (iv) those who only care about recognition and do not care what is done with their work; (v) those who do not care about what is done with their work; and (vi) those who do not wish to share their work. We also need to look at later developers who require resources for their development. They can belong to the following main categories: (i) financially motivated; (ii) recognition motivated; (iii) want to be part of a peer development group; (iv) any combination of the above.

We propose several changes that address the points raised, and discuss each in turn. These changes may either be applied to existing licenses, or may be made a part of a revised open source definition

1. Clarifying open questions under current licenses, which cause reluctance to use OSLs, specifically, improving technical descriptions of derivative work. It is important to realize that technologies are ever-changing and are used in ever-new environments and settings. As such, it is futile to

draft short all-encompassing OSLs. A better solution, we propose, would be to include technical annexes to OSLs that clarify, exemplify and demonstrate their applicability and intent. Such annexes could be part of the license document itself, or could themselves be accessible openly and maintained and updated by the entity proposing the license. For example, for the GPL the Free Software Foundation can document permitted uses and, alternatively, cases where it believes the license was infringed. As border cases arise, these would be added to a 'live' document that address the changing needs. This recommendation is applicable to all OSLs.

2. Adding a uniform solution addressing the contribution and recognition issue by adding a clear requirement to document any and all changes and name the contributors in each individual file. This simple, low-overhead solution will ensure that any contribution to an open source project remains documented and that the contributors receive the proper credit. A simple system of aliases can solve potential privacy problems for developers that are not interested in revealing their identity. By requiring strict documentation accountability will be increased too, resulting in more reliable and better software. This improvement is applicable to all OSLs that do not already require clear contribution documentation. It is particularly important to the non-viral licenses as these are primarily used by developers concerned with reputation rewards and less by those motivated by financial gain.

3. Adding a time dimension to source code release. Under existing OSLs any provisions that govern the release of the code are in effect immediately. A delay could alleviate the fears of for-profit companies that hope to gain financially from selling the software⁴². As our analysis has shown, the motivation of initial developers varies. While many initial developers are not primarily motivated by economic gains, the absence of some minimal profit might deter them from releasing code under OSLs. A delay in the release of the code would allow such initial developers to sell their software during a relatively short period of time (months) and still commit to the principles of open source. By choosing such an option such initial developers can signal to the market that they espouse open source principles and yet maintain the appropriate considerations for their business needs. Moreover, later developers may also agree to subject their code to a license that does not immediately require release of such code. Specifically, such a provision would allow many developers to use code that is now available under OSLs, that they would otherwise not be able to use. The ‘damage’ to the common repository by the delay in source code release seems negligible when compared to the amount of potential code that would be added that would otherwise be released under proprietary license.

4. Continuing the last point, we also recommend establishing a mechanism for for-profit source code trading while the source code is not yet

⁴² This idea is discussed at length by Asay. See note 40 *supra*.

released to the public. As we have seen, just like in the case of music, the key to quality software is access. A radio station can play any song, but has to pay royalties. In the U.S, for example, such royalties are set by Congress and not by the artists whose work is used. Thus, access and compensation become two separate functions. In our case, we recommend a provision to OSLs that would make releasing the source code a requirement under certain conditions, but for an extra payment that would be set not according to varying market conditions but by an objective body. This is very different from the situation today under which the market alone sets the price for source code creating insurmountable transaction costs⁴³. We can imagine an entity like ICAN (the organization that governs internet domain distribution) in such a source-code pricing role. This provision is only relevant in cases where the software is released under an amended OSL, it does not apply to proprietary software release under a closed-source license.

5. Establishing the parallels of Fair Use. Today, the appropriation of any amount of code released under an OSL is considered an infringement of that OSL (particularly if that code is used in a closed-source application). This limitation hinders close coupling of open and closed software projects. Extending the rationale that allows Fair Use of copyrighted material for purposes of satire or critique, we suggest that this situation can be ameliorated by allowing a limited amount of open source code to

⁴³ e.g. in the extreme: if someone wanted Window's code so badly, they could potentially buy Microsoft, but this event is not likely

be used in a closed source environments, provided that this use is intended to facilitate better interfacing between open source and proprietary software. This provision is particularly relevant to the GPL which is the strictest license when it comes to the terms of combining GPLed and non-GPLed software.

Taken together, these provisions suggest limiting the number of available licenses by refining the definition of Open Source. Based on the points above, a new definition of OSLs can be derived that takes into account the maximization of innovation as one of its goals. By applying these improved standards simple and effective open source licenses can be released that address the needs of the different players that participate in the world of open source.

7 Conclusion

This paper has developed an analytical framework that enabled us to inspect the significant effect that OSLs have on innovation, and suggest improvements that would maximize it. By drawing distinctions between 'initial' and 'later' developers, and using an in-depth analysis of the positive and negative effects that certain OSL provisions have on innovation, we were able to better account for the diverging motivations of developers and would-be innovators, and explicate the problems that persist in existing OSLs.

The focus of our analysis has been the possibility that certain changes to exiting OSLs or to the open source definition might positively impact innovation. By analyzing OSLs within their social context, and demonstrating their historical role as a foundation to the open source model, we were able to argue that to the extent that such changes will strike the appropriate balance among developers' diverse motivations, the likely outcome is higher measures of innovation. We concluded with a set of practical recommendations, which, while not exhaustive, addresses the key problems of imbalance that we have identified.

Potential objections to our argument might come from several directions. First, one might attack the analytical distinction we make among temporal classes of developers. While we acknowledge that in the knotty reality of open source development a developer rarely fits nicely into such clear-cut categories we believe that the practicalities of this situation do not undermine our key conclusions. Although a developer (individual, community, or corporation) can be part of both the 'initial' or 'later' developer group at different times or even simultaneously (while pursuing different projects or different parts of the same project), the considerations that are applicable to license choice decisions remain intact.

Secondly, an objector might attack our assumption that increasing the common repository is a forerunner of innovation. Here we would answer that, based on our analysis of the positive effects of OSLs on innovation it should be clear why this assumption is justified. While measuring innovation is a tricky task, there is substantial reason to believe that maximizing the common repository of code while ensuring proper rights to later developers will in fact encourage innovation, as our argument has showed.

Thirdly, one might question our very rationalization about the need to account for the motivations of for-profit entities that have hitherto refrained from partaking in the open source model. One might ask why these players have not used the permissive licenses that would have allowed them to enjoy the benefits of the open source model without paying the full price for it. To answer this objection we need to remember that software development is an ongoing process. The key reason that, save for a small number of counter-examples, the more permissive licenses have been largely abandoned (and the GPL has become a de-facto standard) is that such licenses failed to attract a critical mass of (initial) developers that would be willing to contribute to an ongoing software development effort. Only by maintaining the strictness of the GPL on one hand, but accepting the appropriate modifications, as outlined above, on the other hand, can one hope to

garner to full force of the open source community and combine it with the traditional innovative forces in the software industry.

As our analysis has shown, OSLs play a significant role in the intricate reality of open source development. As the open source phenomenon matures, it is important that we address the implications it may have for the interaction of the law, technology, and society. Focusing on innovation and its maximization is a prime example of this line of research.