
From Software to Life Sciences: The Spreading of the Open Source Production to New Technological Areas

Joseph Eng, Jr., Ph.D.*

I. INTRODUCTION

Open source software development is a method of software production in which a programmer releases a program to the public with a license that gives users the right to use the program for any purpose, to modify it, and to distribute the original or modified program without paying royalties.¹ Once regarded as a curiosity or a temporary phenomenon, open source software development has become a permanent and growing part of the landscape of software development. In many commercial settings, the functionality and stability of software developed using the open source method matches or exceeds that of the proprietary software, allowing open source software to compete successfully against proprietary software. For example, in the web server market, the open source program known as “Apache” has captured approximately 70% of the web server market share as of January 2005,² and continues to gain market share at the expense of all other competitors, including

* Scientific Advisor at Morgan & Finnegan. Ph.D., Physical Chemistry, Columbia University; candidate for *Juris Doctor*, Fordham University.

¹ See David A. Wheeler, Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers! (Nov. 14, 2005), http://www.dwheeler.com/oss_fs_why.html (analyzing quantitative measures of the equality or even superiority of the open source model to that of proprietary software development).

² *Id.* Even more impressive is that Apache’s market share is over three times that of the second-place competitor. *Id.*

those that produce proprietary software.³ In the operating system market, the open source software known as “Linux” has made spectacular gains in market share over a short time.⁴ As of June 2001, Linux was running on approximately 30% of the machines connected to the web,⁵ up from a mere 0.1% as of May 1999.⁶ Furthermore, the open source internet browser known as Firefox, though currently possessing less than 10% of the browser market, appears to be rapidly eroding the market dominance of Microsoft’s Internet Explorer.⁷

Other technological fields, taking notice of the stunning successes in open source software development, have also proposed production models based on the “open source” philosophy. For example, “open source” production has been reported in bioinformatics,⁸ genomics,⁹ electronic circuit design,¹⁰ hardware design,¹¹ and computer processor design.¹² Moreover, within the last year, open source production has been proposed for development of biotech methods for gene manipulation,¹³ for development of medicines,¹⁴ and for database software.¹⁵

Given the proliferation of the open source production model to so many different industries, one may be tempted to speculate that it can be readily adopted in any industry. However, the “share and share-alike” mentality that pervades the open source software movement is not the norm in every industry. In fact, some members

³ *Id.*

⁴ See Wheeler, *supra* note 1, at §§ 2.2-2.4 (describing the extent of market penetration of the Linux operating system).

⁵ *Id.*

⁶ Market Share of Computers & OS, <http://www.jmusheneaux.com/O-MARKET.htm#25> (last visited Feb. 9, 2006).

⁷ Due to recurring security problems, Microsoft’s Internet Explorer is rapidly losing market share for internet browsers. For example, Explorer has lost one percent of the internet browser market in only one month. Wheeler, *supra* note 1.

⁸ *Symposium on Bioinformatics and Intellectual Property Law*, 8 B.U. J. SCI. & TECH. L. 254 (2002); JONATHAN DUGAN, OPEN SOURCE INITIATIVES IN BIOINFORMATICS (Aug. 2001), http://www-smi.stanford.edu/pubs/SMI_Reports/SMI-2001-0902.pdf.

⁹ Alexander K. Haas, *The Wellcome Trust’s Disclosures of Gene Sequence Data Into the Public Domain & Potential for Proprietary Rights in the Human Genome*, 16 BERKLEY TECH. L.J. 145 (2001).

¹⁰ Electronic Design Automation (EDA) Project, <http://www.geda.seul.org> (last visited Feb. 9, 2006).

¹¹ John G. Spooner, *Open-source Credo Moves to Chip Design*, CNET News.com, Mar. 27, 2001, <http://news.com.com/2100-1001-254816.html>; Jamil Khatib, *Open Hardware Design Trend* (Jan. 19, 2004), <http://www.opencores.org/articles.cgi/view/12>.

¹² Peter Clarke, *Momentum Builds for Open-Source Processors*, EE TIMES, Feb. 1, 2001, <http://www.eetimes.com/story/OEG20010201S0050>; Nicholas G. Lesniewski-Laas, *Open Source Initiative Centers Around IBM’s Power Architecture*, CHIPGEEK, Dec. 2, 2004, <http://www.geek.com/news/geeknews/2004Dec/bch20041202028080.htm>; Power Architecture Technology, <http://www.power.org> (last visited Feb. 8, 2006) (“Power.org’s mission is to develop, enable and promote Power Architecture technology as the preferred open standard hardware development platform for the electronics industry and to administer qualification programs that optimize interoperability and accelerate innovation for a positive user experience”).

¹³ Andrew Pollack, *Open-Source Practices for Biotechnology*, N.Y. TIMES, Feb. 10, 2005, at C8.

¹⁴ *An Open-Source Shot in the Arm?*, 371 ECONOMIST 8379, June 10, 2004 [hereinafter ECONOMIST]; Stephen M. Maurer, Arti Rai & Andrej Sali, *Finding Cures for Tropical Diseases: Is Open Source an Answer?*, 1 PLOS MED. 183, available at http://medicine.plosjournals.org/archive/1549-1676/1/3/pdf/10.1371_journal.pmed.0010056-S.pdf.

¹⁵ Ken Spencer Brown, *Like Linux, Databases Going Open Source; Traditional Sellers Face Threat; IBM Oracle and Microsoft Could See Customers Adopt Free Open-Source Software*, INVESTORS BUS. DAILY, Feb. 7, 2005, at A5.

of the largest industries have consistently displayed the opposite behavior, either by aggressively using intellectual property rights to block the use of their products by others, or by hoarding knowledge to maintain a competitive advantage. For example, previous studies have shown that semiconductor manufacturing companies rely on secrecy, lead time, and unique manufacturing or design capabilities to maintain competitive advantages.¹⁶ Such practices are clearly inconsistent with an open source production model, which favors permitting others to use one's technology, as well as prompt public disclosure of any new advances.

This paper explores whether open source production is universally applicable schema for innovation, and examines some of the legal and economic factors that are a prerequisite for a given industry to adopt an open source production model. In particular, this paper compares open source software production, which is already firmly established, with two fledgling open source movements in the life sciences that have been recently proposed but not yet widely adopted. From these comparisons, two important themes emerge. First, any useful conceptual framework for understanding the underlying legal structure of the open source production model must take account of whether an industry relies on patents as the primary method of enforcing intellectual property or on copyrights. This has important implications, not just for issues involving the enforcement of intellectual property rights, but also for determining who can participate in open source production. A second recurring theme concerns the ability of a potential contributor to an open source movement to externalize costs, particularly those related to production and to enforcement of intellectual property rights. As discussed below, the inability to externalize costs can provide strong disincentives to a would-be contributor, even if strong personal reasons exist for making an open source contribution.

Part II of this paper examines open source production in software as a paradigm, focusing on the personal incentives that motivate open source contributors, as well as the mechanisms for preserving open source production, such as open source licenses. This analysis provides a basis for a comparison in Part III with open source movements in crop engineering and drug development. Here, it will become apparent that the contours of an open source production model for copyrightable subject matter (e.g., software) must be qualitatively different than those of an open source model for patentable subject matter (e.g., pharmaceuticals). Unique issues exist as to patentable subject matter that are not present in the open source production of copyrightable subject matter. For example, in many cases, the ability of a researcher in biotechnology to externalize research costs, such as those related to equipment, reagents, and other scarce resources, will be a critical factor in determining whether a researcher can contribute to open source production. Additionally, serious problems may arise when one attempts to enforce open source licenses concerning subject matter that, because of its nature, can only be protected by patent law. Enforcement of an open source license in such cases is particularly difficult for inventors who, because of a lack of capital and/or legal expertise, are unable to secure patent protection for their inventions.

¹⁶ Bronwyn H. Hall & Rosemarie Ham Ziedonis, *The Patent Paradox Revisited: An Empirical Study of Patenting in the U.S. Semiconductor Industry, 1979-1995*, 32 RAND J. ECON. 1, at 101-28 (Spring 2001).

II. OPEN SOURCE SOFTWARE PRODUCTION

A. What is Open Source Software?

The best way to understand open source software is to contrast it to conventionally produced commercial software. Under the conventional software production model, when a programmer working for a corporation writes a computer program, he or she normally chooses a programming language (e.g., C++) and proceeds to write a series of commands in that programming language to achieve a certain function (e.g., to send e-mail). This series of commands is known as the “source code” and, in principle, could be freely edited by anyone who has access to it and permission to do so.¹⁷ One may be motivated to edit the source code of a program for many reasons. Returning to the e-mail program example, one might edit the source code to include new functions (e.g., a capability to handle attached files), to improve compatibility with emails originating from different machines, or simply to fix earlier programming errors (i.e., “to debug”). When the programmer feels that the editing of the source code has been completed, the programmer “compiles” the program by using software tools to transform the source code into an “object code” (also known as “executable code”). Compared to the source code, the object code is nearly indecipherable to humans, but much easier for a machine to read.¹⁸ The object code is then recorded on a storage medium, such as a CD, which is subsequently packaged and sold to the public. Under conventional notions of proprietary software, it is advantageous for a software company to sell only the object code to the consumer, because the object code is not particularly amenable to reverse engineering by others, such as competitors.

For most people, this arrangement is acceptable. Most users of commercial software lack the expertise and/or desire to modify the source code of a program to create new functions, therefore the failure of software manufacturers to provide the source code with the commercial product is not a problem. These users purchase software based on the functions built into the software by the software manufacturer. Thus, when a particular software product lacks a desired function, the user simply chooses a competing product that has it, rather than asking the software manufacturer for the source code in order to modify the program. Similarly, when a particular software product contains programming errors (“bugs”), rather than debugging the originally purchased program, the average user will simply learn to live with the bugs or will replace the program with that of a competitor.

In direct contrast, when a user obtains open source software, both the executable code and the source code are provided under an open source license.¹⁹ While the actual details of the open source license may vary, as discussed below, generally the license will permit unrestricted copying, modification, and redistribution of the software, provided that subsequent versions or copies of the program are similarly

¹⁷ 17 U.S.C. § 102 (2005) (Original computer programs are protected by copyright once they are “fixed in a tangible medium of expression.” Accordingly, if one were to misappropriate a program at this stage and use it for his own purposes, the programmer could bring an action for copyright infringement.)

¹⁸ See Stephen M. McJohn, *The Paradoxes of Free Software*, 9 GEO. MASON L. REV. 25 (2000) (providing a lucid discussion of the differences between source code and object code).

¹⁹ Natasha T. Horne, Note & Comment, *Open Source Software Licensing: Using Copyright to Encourage Free Use*, 17 GA. ST. U. L. REV. 863, 871-73 (2001).

protected under an open source license. Often, the license will also have additional clauses, such as an “attribution clause,” which requires improvements to the program to be properly attributed to the responsible programmer.²⁰ Also, unlike most commercial software, open source licenses often come with express disclaimers of fitness and implied warranty of merchantability, which help to encourage open source contributions by shielding the contributing programmers from liability.²¹

Many legal scholars, economists, and others have noted that open source development of software can be a superior method of production.²² Some of the key advantages of using an open source production model rather than a proprietary production model to develop software are as follows:

(1) Open source software is usually more robust: The robustness of open source software results from at least three effects. First, because an open source community is significantly larger than the community of programmers at proprietary software firms, it will likely contain a greater number of talented programmers. Second, more people study the source code when software is developed using an open source model, so that bugs are found and corrected more quickly.²³ Third, because several independent programmers or groups of programmers may work on a given problem, multiple solutions to the problem are presented. In this way, the different solutions may be compared against each other and the best solution adopted.

(2) Open source software often has more functions: When a proprietary software firm develops a program, it may intentionally omit certain functions because the cost of adding them would be greater than the corresponding economic rewards, or because it wants to compel the market to evolve. For example, a proprietary design firm may choose to develop software that works with the predominant platform that exists in the market, even though there is a sizable group of people using an alternative platform. In such situations, the users of the minority platform are usually forced to switch to the majority platform if they want to run that particular software. In contrast, when software users have the source code, they may add new functionalities to the program as needed. This may even include periodically modifying a program to avoid obsolescence.²⁴

²⁰ *Id.* at 873.

²¹ GNU General Public License, <http://www.gnu.org/copyleft/gpl.html> (last visited Feb. 9, 2006).

²² See Wheeler, *supra* note 1 (stating that users create incredible value through the open source software development process by tinkering with the product); McJohn, *supra* note 18, at 66 (citing openness as a “great virtue”); Horne, *supra* note 19, at 866 (stating that through collaboration open source products develop quickly and efficiently). See also Josh Lerner & Jean Tirole, *The Simple Economics of Open Source*, 10 (Dec. 29, 2000), <http://www.hbs.edu/research/facpubs/workingpapers/papers2/9900/00-059.pdf> (stating that open source software development dominates traditional development, particularly among sophisticated users); Marcus Maher, *Open Source Software: The Success of an Alternative Intellectual Property Incentive Paradigm*, 10 FORDHAM INTELL. PROP. MEDIA & ENT. L.J. 619 (Spring 2000) (stating that “open source software attains high technical standards”); Mark Haynes, *Commentary: Black Holes of Innovation in the Software Arts*, 14 BERKELEY TECH. L.J. 567, 575 (1999) (arguing that the closed source copyright method of software development creates a technical “black hole” and is bad for innovation).

²³ “[G]iven enough eyeballs, all bugs are shallow.” Eric Steven Raymond, *The Cathedral and the Bazaar* Release Early, Release Often, (2002), <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html>.

²⁴ Ardour, <http://ardour.org/money.php> (last visited Feb. 9, 2006) (For example, Ardour, producer of digital audio workstations, advertises on their web site that their products will never become obsolete because they use open source software, which can be upgraded and improved as technology improves).

(3) Open source software evolves more quickly: It is costly for proprietary software firms to develop new editions of programs. Thus, proprietary software firms typically avoid doing so until the market forces absolutely require it. For example, it has been noted that Microsoft has been extremely slow to develop new versions of Internet Explorer once it vanquished the rival Netscape browser and attained a monopoly in the internet browser market.²⁵ In contrast, open source code is upgraded whenever one of the users at large perceives a need to do so and writes the appropriate code.²⁶

B. Understanding the Motivations for Open Source Software Development: The “Carrot or Stick” Model

At first glance, the willingness of programmers to give away their programs under the open source production model is puzzling, because it appears to contradict traditional economic incentive theory. This theory states that individuals will not invent or create unless the expected returns outweigh the invention or creation costs.²⁷ The expected returns may be tangible, such as a monetary reward, or intangible, such as a competitive advantage in a market.²⁸ Producing software under an open source production model, however, does not appear to yield either type of benefit; often an open source programmer gives away his or her programs for free, and anyone (including competitors) has free use of the programs, thereby negating any advantage that the programmer otherwise would have gained. Thus, to understand the private motivations for open source production, one must dig deeper.

Before considering the extensive literature concerning the incentives for contributing to the open source software movement, it is useful to adopt a classification scheme to provide a conceptual framework. One such classification scheme is the “carrot or stick” model, which is named after the age-old method of driving a donkey cart using alternately a carrot (placed in front of a hungry donkey to motivate him to walk forward) or a stick (forcibly applied to the rump of the donkey, if the carrot fails). In the context of the open source software movement, incentives that fall in the “carrot” category are those that appeal to the programmer’s own desires, such as public recognition of programming prowess, as discussed below. On the other hand, “stick” incentives are those that prevent programmers who are considering violating the open source credo (e.g., by attempting to take credit for someone else’s work) from doing so under the pain of legal action or retribution for violation of societal norms. As discussed below, this “carrot or stick” classification is also useful for understanding non-software based open source movements, as well. Accordingly, the following subsections discuss the major “carrot” and “stick” reasons that have been set forth to justify open source software

²⁵ Haynes, *supra* note 22, at 574.

²⁶ Yochai Benkler, *Coase’s Penguin, or, Linux and the Nature of the Firm*, 112 YALE L.J. 369, 376 (2002) (describing this phenomenon as a situation where “very large aggregations of individuals independently [scour] their information environment in search of opportunities to be creative in large or small increments”). The individuals then select appropriate tasks for themselves and perform them for a variety of reasons.

²⁷ See Mark A. Lemley, *The Economics of Improvement in Intellectual Property Law*, 75 TEX. L. REV. 993 (1997) (stating that an inventor will not invest in an invention or creation unless the expected return outweighs the costs of doing so).

²⁸ *Id.*

production.

1. *“Carrot” Incentives in Open Source Software Development*

As previously noted, most open source programmers do not get paid for distributing their programs. That is not to say, however, that they do not accrue any benefits from doing so. Lerner and Tirole have suggested that “signaling incentives” often provide strong motivation for individual programmers to distribute their programs as open source code.²⁹ Here, the term “signaling incentives” refers to the desire of a programmer to become well known through the publication of his or her work. The reward consists of two parts: (1) financial return, which may derive indirectly from the advantages gained through publication, such as increased possibility of future job offers, a better chance to obtain venture capital funding, or possibly shares in a start-up software company; and (2) ego gratification, which relates to the desire to be recognized by fellow programmers.³⁰ The important role of ego gratification through peer recognition is the reason that attribution clauses are included in many open source software licenses.³¹

In addition to signaling incentives for individual programmers, Lerner and Tirole have also considered the motivations for a commercial company to release otherwise proprietary code as open source code.³² Their analysis is based on the assumption that companies are rational economic actors, and that companies will make source code “open” only if the following two conditions are met: (1) there is an increase in profits in another part of their business, and (2) that profit increase offsets any gains that would have resulted had the source code not been released.³³ According to Lerner and Tirole, “the temptation to go open source is particularly strong when the company is too small to compete commercially in the primary segment or when it is lagging behind the leader and about to become extinct in that segment.”³⁴ Such reasoning may help to explain, for example, the reason that IBM decided to support the open source Linux operating system after several costly and unsuccessful attempts to wrest control of the operating system market from Microsoft.³⁵

Other commentators have pointed to a “hacker ethic” as a motivational factor for open source contributors. The “hacker ethic” refers to a set of norms that are meant to govern the behavior of computer programmers, primarily those who do not belong

²⁹ JOHN LERNER & JEAN TIROLE, *THE SIMPLE ECONOMICS OF OPEN SOURCE* 18 (Feb. 25, 2000), <http://www.hbs.edu/research/facpubs/workingpapers/papers2/9900/00-059.pdf>.

³⁰ *Id.* In addition, others have recognized the importance of increased social status to open source software contributors. See, e.g., McJohn, *supra* note 18, at 42 (stating that software developers look to increase their status among their peers by “showing off technical prowess, or receiving approval for participating in the open source movement, or building relationships in the development process”).

³¹ See BSD License, available at <http://www.opensource.org/licenses/bsd-license.html> (last visited Feb. 9, 2006) (providing a template for software developers who wish to create their own attribution clauses).

³² Lerner, *supra* note 29.

³³ *Id.*

³⁴ *Id.*

³⁵ See Dennis E. Powell, *The Sad Parable of OS/2* (Sept. 1997),

<http://www.linuxandmain.com/features/os2retro.html> (discussing IBM’s unsuccessful attempt to create an operating system). See also Stephen Shankland, *IBM: Linux Investment Nearly Recouped*, CNET NEWS.COM, Jan. 29, 2002, <http://news.com.com/2100-1001-825723.html> (noting that IBM invested one billion dollars in Linux in 2001, but managed to recoup that investment after only one year).

to institutions that generate proprietary software. Two of the central tenets of the hacker ethic are “information wants to be free” and “mistrust authority – promote decentralization.”³⁶ Followers of the hacker ethic believe that such ideals will, in the long run, benefit society by empowering ordinary citizens against oppressive governments or corporations. Thus, the hacker ethic contains a strong undercurrent of societal altruism mixed with anti-establishment sentiment. Accordingly, many programmers who follow the hacker ethic have embraced open source software production, because the decentralized, distributive nature of open source software development is consonant with their ideals.

Eric S. Raymond, one of most well known proponents of the open source software movement (and the one who popularized the term “open source”), has focused on another aspect of hacker culture to help explain the open source software movement. He describes the hacker culture as a “gift culture,” where the norms are significantly different from those in most other cultures.³⁷ According to Raymond, the norms of a “gift culture” are based on abundance, rather than scarcity, of survival goods, so that normal commercial transactions (e.g., the exchange of goods) are “an almost pointless game.”³⁸ Thus, rather than basing social status on material acquisition or power to control scarce resources, as do most societies, “social status [in a gift culture] is determined by . . . what you give away.”³⁹ This would help to explain, for example, why major contributors to the open source software movement, such as Linus Torvalds, are greatly admired within the open source community.

Starting with the diverse motivations for open source software production, Yochai Benkler has extrapolated the theoretical underpinnings of the open source model to explore the model’s potential application in other fields of information generation.⁴⁰ Benkler argues that the various personal motivations of an individual engaging in “commons-based peer production of information” (e.g., open source software production) can be unified and understood by considering the total reward that the individual receives as a result of the contribution.⁴¹ The total reward consists not only of a monetary reward, but also an intrinsic hedonic reward (i.e., personal pleasure associated with making the contribution) and a social-psychological reward (i.e., improvement in social relations and status).⁴² Benkler explains that the interplay between these three components that constitute the total reward is important in determining whether an individual will participate commons-based peer

³⁶ Other aspects of the hacker ethic include the belief that computer programming is a form of personal expression and that programming is sheer joy. As Linus Torvalds (the creator of Linux) puts it: “the computer itself is a form of entertainment.” PEKKA HIMANEN, LINUS TORVALDS & MANUEL CASTELLS, *THE HACKER ETHIC* (2001). See also *Free Software: Hackers Comeback*, <http://www.free-soft.org/softwarewar.html>.

³⁷ Eric Steven Raymond, *The Hacker Milieu as Gift Culture from Homesteading the Noosphere*, <http://www.catb.org/~esr/writings/homesteading/homesteading/ar01s06.html>.

³⁸ *Id.* This “abundance” exists because there is “no shortage of disk space, network bandwidth, or computing power.” See Rishab Aiyer Ghosh, *Cooking-Pot Markets: An Economic Model for ‘Free’ Resources on the Internet* (Dec. 7, 1999), <http://www.heise.de/tp/r4/artikel/6/6432/1.html> (last visited February 6, 2006) (noting that the costs of copying a digital creation, even a million times, is negligible and spread over as many users, which may be seen as adding to the definition of “abundance”).

³⁹ HIMANEN, *supra* note 36.

⁴⁰ Haynes, *supra* note 22.

⁴¹ *Id.* at 426.

⁴² *Id.*

production of information.⁴³ In particular, an individual “will pursue an activity that has low, zero, or even negative monetary rewards when the total reward, given the hedonic and social-psychological rewards, is higher than alternative courses of action that do have positive monetary rewards attached to them.”⁴⁴

2. “Stick” Incentives: Enforcement of the Open Source Credo through Laws and Norms

In direct contrast to “carrot” incentives, which motivate a programmer from within to adopt the open source production model, “stick” incentives are external to the programmer and are designed to punish programmers who violate the open source credo. Generally, misbehavior involves misappropriation in various forms, such as taking freely available open source code and trying to fence it in (e.g., by patenting it and then asserting the patent),⁴⁵ or taking credit for open source code that someone else wrote. Misappropriation is a serious threat to the continued viability of the open source software community because, when rampant, it provides strong disincentives to future contributions by community members.

Generally, there are two primary types of “stick” incentives for deterring misappropriation: (1) enforcement through norms, and (2) enforcement through law. Under a norms-based enforcement approach, when bad actors violate the norms of a community, other members of the community punish them, rather than the government.⁴⁶ As Professor Arti Rai explains, “[s]anctions imposed on norm violators may include everything from informal gossip and disdain to formal exclusion from the group governed by the norms.”⁴⁷ As an example of norms-based enforcement at work, consider the case where patentees sought to assert patents against an open source defendant that were directed to certain fundamental multimedia techniques, fixing the Y2K bug, and a privacy protection algorithm.⁴⁸ The actions of the patentees caused considerable outrage in the open source community, because the subject matter of the patents was not considered to be novel within the community, and was viewed as the patentees’ attempt to fence in free, publicly available code.⁴⁹ As a result, several members of the open source community searched for and submitted prior art to the U.S. Patent and Trademark Office that could be used to invalidate the patents-in-suit.⁵⁰

The second “stick” incentive, namely enforcement through law, is grounded in intellectual property rights, and in particular copyright and licensing. Under U.S. copyright law, when a programmer fixes an original computer program in a tangible

⁴³ *Id.*

⁴⁴ *Id.* at 429.

⁴⁵ GNU General Public License, *supra* note 21, at 51.

⁴⁶ Arti Kaur Rai, *Regulating Scientific Research: Intellectual Property Rights and the Norms of Science*, 94 NW. U. L. REV. 77 (1999) (discussing the importance of norms in governing behavior of communities).

⁴⁷ Note that a norms-based approach has its limits. Citing game theoretic models of behavior, Professor Rai states that “cooperative norms will emerge only if the players are small in number; have repeated long-term interactions; can readily observe each other’s moves; and do not discount the future too heavily.” *Id.* at 85.

⁴⁸ McJohn, *supra* note 18, at 51.

⁴⁹ *Id.*

⁵⁰ *Id.*

medium of expression, copyright protection attaches to the program, preventing others from making or distributing unauthorized copies, or from preparing unauthorized derivative works based on the original program.⁵¹ Normally, the ability of the original programmer to exclude use by others under copyright law induces those who wish to use the program to obtain a license. Generally, the license limits the user's ability to copy and distribute the program, and often will require the user to pay royalties for the right to use the program.

Open source software licenses, however, use the exclusion right inherent in copyright law differently.⁵² Rather than using the exclusion right to extract a monetary payment or to restrict distribution or modification of the program, an open source software license typically uses the exclusion right to ensure that a program, as well as subsequently modified versions of it, remains freely available to others as open source code.⁵³ In addition, the license may require a user to follow other rules that reflect the values of the open source software community, such as properly attributing authorship (which is important for the purposes of signaling, as discussed above),⁵⁴ not using the open source code in commercial or proprietary software,⁵⁵ or shielding the original programmer from liability via an "as is" clause.⁵⁶ Enforcement of these terms does not rely on the threat of a legal action for breach of contract, as is the case for most license agreements. Rather, because the original programmer has reserved his copyright and bundled it with the licensing terms, he may bring a copyright infringement action against non-compliant licensees. This inverted use of copyright protection (i.e., allowing copying, distribution, and modification of a program by others, subject to the open source licensing terms) is playfully called as "copyleft" protection within the open source software community.⁵⁷

"Stick" incentives, and in particular strong intellectual property rights, are an integral part of the continued viability of an open source movement. As discussed in the next section, where no such "stick" incentives exist, it is much more difficult to prevent opportunistic behavior that can threaten to derail open source production.

III. OPEN SOURCE PRODUCTION IN THE LIFE SCIENCES

As noted in the introduction, other technological fields have embraced the open source production model.⁵⁸ Two recently proposed open source movements in the

⁵¹ See 17 U.S.C. § 102 (discussing subject matter of copyright).

⁵² See Natasha T. Horne, Note & Comment, *Open Source Software Licensing: Using Copyright to Encourage Free Use*, 17 GA. ST. U. L. REV. 863, 877-889 (2001) (discussing the various terms of these licenses). Currently, many different open source licenses are available for protecting open source software. Particularly well known ones include the GNU General Public License ("GNU GPL"), the Berkeley Software Distribution License ("BSD"), the Aladdin License, the Mozilla Public License ("MPL"), the Netscape Public License ("NPL"), and the Sun Community Source License ("SCSL").

⁵³ GNU General Public License, *supra* note 21.

⁵⁴ BSD License, <http://www.opensource.org/licenses/bsd-license.html> (last visited Feb. 9, 2006).

⁵⁵ Debian, Software in the Public Interest, Inc., What Does Free Mean?, <http://www.debian.org/intro/free> (last visited Feb. 9, 2006).

⁵⁶ Version 1.1 of the Mozilla Public License, <http://171.65.65.92/repos/protégé/jambalaya/trunk/ca.uvic.cs.chisel.jambalaya/license.html> (last visited Feb. 9, 2006).

⁵⁷ GNU Project, What is Copyleft?, <http://www.gnu.org/copyleft/> (last visited Feb. 9, 2006).

⁵⁸ Strictly speaking, it is a misnomer to use the term "open source" to describe production methods in technologies unrelated to software development. This is because term "open source" implicitly refers to

life sciences are open source crop engineering and open source drug development. Because both open source movements are in their infancy, many of their characteristics are still evolving. Thus, in this analysis, it will be difficult at times to draw comparisons with established open source movements, such as open source software development. Nevertheless, from the current understanding of these two new open source movements, it is already evident that the “carrot” and “stick” incentives discussed above in connection with open source software development are insufficient for providing a full description of the motivations here. Before examining these differences, however, a discussion of the contours of each of these open source movements is useful to provide context.

A. Example 1: Open Source Development of Crop Engineering Methods

The process of genetically engineering plants typically involves inserting foreign genetic material (a process known as “transfection”) into a target plant, in order to produce certain desirable plant properties (e.g., resistance to pests). Although there are several possible ways to introduce genetic material into a plant, one of the most efficient ways involves using a soil microbe known as *Agrobacterium Tumefaciens*.⁵⁹ When this bacterium infects certain types of plants, it causes a crown gall (i.e., a tumorous lump) to form at the site of infection.⁶⁰ The crown gall results from the incorporation of a portion of the bacterium’s genetic material (known as the “tumor-inducing plasmid”) in the genome of the host plant.⁶¹ By using certain genetic engineering techniques, the mechanism by which the tumor-inducing plasmid becomes incorporated within a plant genome can be exploited to incorporate foreign genes that are specifically chosen by the research scientist for their ability to confer desirable plant properties.⁶²

The technology of using *Agrobacterium Tumefaciens* as a vehicle for introducing genetic material into crop plants is controlled by numerous patents belonging to Monsanto, a leading supplier of genetically engineered crops.⁶³ Prior to 2005, *Agrobacterium Tumefaciens* was the only known microbe with the ability to introduce foreign genetic material into target plants. However, in the February 10, 2005 issue of *Nature*,⁶⁴ an Australian organization known as the Center for Application of Molecular Biology to International Agriculture (“CAMBIA”) published a paper that provided three different strains of bacteria that could serve as

the source code of a computer program. Thus, for technologies unrelated to software development, there is no “source code” to speak of. Putting aside semantics, however, we will define “open source production” in technologies that do not involve software development as referring to production which ultimately generates a product (tangible or otherwise) that can be freely used, modified, and/or redistributed, just as software is in the open source software production model.

⁵⁹ Besides using *Agrobacterium Tumefaciens*, other methods of introducing genetic material into plants involve techniques such as microinjection, electroporation, and particle bombardment. See, e.g., Access Excellence, Transforming Plants – Basic Genetic Engineering Techniques, http://www.accessexcellence.org/RC/AB/BA/Transforming_Plants.html (describing the cloning of plant cells and the manipulation of plant genes).

⁶⁰ Clarke, *supra* note 12; Lesniewski-Laas, *supra* note 12.

⁶¹ *Id.*

⁶² *Id.*

⁶³ Latha Jishnu, *Now, Open Source in Biotech*, BUS. WORLD, Mar. 7, 2005, at 46.

⁶⁴ Wim Broothaerts, et al., *Gene Transfers to Plants by Diverse Species of Bacteria*, 433 NATURE 629 (Feb. 10, 2005), available at <http://www.bios.net/daisy/bios/393/version/live/part/4/data>.

alternatives to *Agrobacterium Tumefaciens*.⁶⁵ Rather than patenting this technology, as Monsanto had done with *Agrobacterium Tumefaciens*, CAMBIA decided to use this technology as a starting point for launching an open source initiative known as the Biological Innovation for Open Society (“BIOS”).

The BIOS initiative is directed to three interdependent activities: (1) promoting open source development of biotechnology, (2) developing new legal mechanisms for protecting open source innovations, and (3) providing intellectual property informatics and analysis to help researchers avoid patent liability.⁶⁶ For the purposes of this paper, we focus on the first two activities, although the importance of the third activity, IP informatics, should not be underestimated, given the complexity of the patent landscape of the biotechnology field.⁶⁷

With respect to the open source development of biotechnology, BIOS proposes an open source production model analogous to that of the open source software movement. This model includes, for example, the organization of open source communities through the use of a centralized webpage for coordination and publication of projects.⁶⁸ However, rather than limiting the contributions to only open source software, BIOS focuses on developing portfolios of open source biotech as alternatives to patented portfolios of technology that are “presenting real bottlenecks to innovation by and for the developing world.”⁶⁹ To this end, BIOS has announced two portfolios of open source biotech inventions. The first portfolio, entitled “Genetic Resource Analysis” is concerned with the conservation, identification, and use of genetic resources.⁷⁰ To seed this portfolio, CAMBIA has placed into the portfolio its own patented genotyping technology known as “Diversity Array Technology” (“DArT”TM),⁷¹ an economical, high throughput method that has already proven useful for the molecular breeding of wheat, barley, apples, rice, cattle and sheep. The second portfolio, entitled “Crop Molecular Enabling Technologies,” includes technologies that are directly applicable to crop development. Here, in addition to the non-*Agrobacterium* method of plant modification described above, the portfolio includes a reporter/marker gene technology (called “GUS”) that is useful for studying promoter activity in different plant tissues or plant development stages.⁷² Like DArTTM, the GUS technology is also patented.⁷³

The second prong of the BIOS initiative concerns developing legal mechanisms for protecting open source biotechnology inventions. In this connection, CAMBIA

⁶⁵ BIOS Homepage, *Biological Innovation for Open Society*, <http://www.bios.net/daisy/bios/15>.

⁶⁶ CAMBIA BIOS Initiative, *Now, Open Source in Biotech*, BUS. WORLD, Mar. 7, 2005, available at <http://www.bios.net/daisy/bios/510/version/live/part/4/data>.

⁶⁷ *Id.*

⁶⁸ See, e.g., BioForge, A New Community for Biological Innovation, <http://www.bioforge.net/> (last visited Feb. 9, 2006) (describing BioForge and CAMBIA BIOS Initiative).

⁶⁹ See, e.g., BIOS Licenses, Frequently Asked Questions, www.bios.net/daisy/bios/27/399 (providing information for frequently asked questions regarding BIOS Licenses) (last visited Feb. 9, 2006).

⁷⁰ See, e.g., The CAMBIA BIOS Initiative, <http://www.bios.net/daisy/bios/10/version/live/part/4/data> (providing information on the CAMBIA BIOS Initiative) (last visited Feb. 9, 2006).

⁷¹ U.S. Patent No. 6,713,258.

⁷² GUS Technology, http://www.cambia.org/cambia_ip.html.

⁷³ See, e.g., U.S. Patent Nos. 5,268,463; 5,432,081; 5,599,670; 5,879,906; 6,391,547; 6,429,292; 6,641,996.

has proposed an open source license, which is similar in some respects to open source licenses used in software development.⁷⁴ For example, the BIOS license contains provisions that provide the licensee with “free use” (i.e., a non-exclusive, royalty-free right to use the biotech invention),⁷⁵ as well as “free distribution” (i.e., the right to sublicense the invention to others).⁷⁶ Other similarities include a provision that requires the licensee to make any improvements available to other members of the BIOS open source movement,⁷⁷ and a provision that protects the inventors who contribute to the BIOS initiative from liability by requiring the licensee to accept the technology “as-is.”⁷⁸

Despite these similarities, there are also significant differences. Perhaps the most significant is that the BIOS license agreement reflects the BIOS vision of an open source production model that is more centralized than that of the software industry. For example, CAMBIA envisions the formation and collective defense of a “protected commons” where patentable (but not yet patented) inventions can be discussed in confidence among the members of the BIOS initiative.⁷⁹ Moreover, BIOS proposes cost-sharing between licensees in order to maintain a database of the improvements, using subscription fees that are based on the licensee’s ability to pay,⁸⁰ a feature which is not found in the open source production model for software.

Another, more subtle, difference concerns the legal status of the protected material. Whereas open source software licenses grant a licensee the right to use works that are already protected by an intellectual property right (the copyright), the BIOS license, by its terms, is directed to both protected works (patented inventions) as well as unprotected technology in the form of “patent applications, know-how, data, materials, and business, technical, economical and manufacturing information.”⁸¹

B. Example 2: Open Source Development of Medicines

Despite the rapid advances in medicine during the last century, many diseases afflicting mankind have not been cured. For some diseases, the development of a cure is difficult for technological reasons. For example, despite years of continuous research by both private companies and public institutions, a complete cure for AIDS

⁷⁴ As of this writing, the latest version of this license is Version 1.1, which is available at <http://www.bios.net/daisy/license/210>.

⁷⁵ *Id.* at clause 2.1.

⁷⁶ *Id.* at clause 2.5.

⁷⁷ *Id.* at clause 3.1 (granting to CAMBIA a worldwide, non-exclusive, royalty-free, fully-paid license, with the right to sublicense to other BiOS Licensees, under the *Improvement Patents*, for use which is within the scope of the *Licensed Patents*, and a worldwide, non-exclusive, royalty-free, fully-paid license, with the right to sublicense to other BiOS Licensees, to any *Improvements* not protected under the *Improvement Patents*, and any *Technology Data* and any *Improvement Material* provided by BiOS LICENSEE to CAMBIA and necessary to practice *Improvements*).

⁷⁸ *Id.* at clauses 5.1, 5.2 (acknowledging that the IP and Technology and Improvements are still experimental in nature and each party will hold, for all claims, suits, losses, liabilities, damages, costs, fees, and expenses resulting from its gross negligence or wrongful act or omission during or after the term of this Agreement, breach of this Agreement, or use, commercialization, or sublicensing of IP and Technology or Improvements or any product made thereby, the other PARTY harmless).

⁷⁹ BioForge, *supra* note 68.

⁸⁰ BiOS Licenses, *supra* note 69.

⁸¹ *Id.* at Background Section.

has not been found, owing to the ability of the human immunodeficiency virus to mutate.⁸² For certain other diseases, however, the failure to find a cure does not result from insurmountable technological challenges, but rather from economic limitations.

Tropical diseases are an example of this latter class of diseases. Although more than 500 million people (more than one tenth of the world's population) suffer from tropical diseases at any given time,⁸³ pharmaceutical companies have devoted relatively little of their resources to discovering cures for tropical diseases.⁸⁴ This lack of focus has been driven primarily by economics; most people who suffer from these diseases live in third-world countries and cannot pay for the drugs.⁸⁵ Because the cost of developing new drugs is astronomical, pharmaceutical companies have had little economic incentive to commit resources to curing tropical diseases.⁸⁶

Recognizing the convergence between biology and computing, Maurer, Rai and Sali have recently proposed the establishment of a "Tropical Disease Initiative" ("TDI") that adopts an open source production model for the development of tropical disease cures.⁸⁷ This initiative would consist of two related parts. In the first part, researchers would rely extensively on computers to perform calculations and searches to identify compounds that could be useful for treating tropical diseases. Such computer tasks would include searching the genome of parasitic organisms for new drug targets, identifying drug candidates that could bind to the targets, and estimating the efficacy of the drug candidate. The researchers would post these results on a TDI community webpage that would serve to coordinate research. According to Maurer et al, by aggregating the combined efforts of members of the TDI through the webpage, drug candidates could be discovered incrementally, "much the same way that LINUX builds operating systems."⁸⁸

In the second part of the initiative, experimentalists would perform "modest chemistry and biology experiments" to confirm that the proposed drug candidates work. Because several different research groups may openly collaborate on these experiments, the cost to any one group can be significantly less than if that group had to perform all of the experiments alone.

⁸² AIDS.org, AIDS Fact Sheet: HIV Resistance Testing, <http://www.aids.org/factSheets/126-HIV-Resistance-Testing.html> (last visited Feb. 9, 2006).

⁸³ ECONOMIST, *supra* note 14.

⁸⁴ To be fair, progress is being made against tropical diseases by "virtual pharmaceutical companies," which are institutions that do not have physical lab space, but nevertheless manage to identify drug candidates through agreements with commercial and academic institutions. ECONOMIST, *supra* note 14; Maurer, *supra* note 14 (providing more details). However, it is estimated that only about 1% of newly developed drugs are for tropical diseases. *Id.*

⁸⁵ Maurer, *supra* note 14.

⁸⁶ Haas, *supra* note 9, at 148 n.18. Haas has noted that it takes, on average, \$500 million to develop a new drug. This figure includes failed attempts.

⁸⁷ Maurer, *supra* note 14. See BIO 2004, http://www.economist.com/science/tq/displayStory.cfm?story_id=2724420 (noting that the paper authored by Maurer, et al., was presented at the BIO 2004 meeting in San Francisco).

⁸⁸ *Id.*

C. “Carrot” Incentives Revisited

1. *Variations across Different Technologies*

As noted at the beginning of this section, both open source crop engineering and open source drug development are still in their infancy.⁸⁹ For this reason, it is difficult to assess with certainty which personal motivations (“carrot” incentives) will play a major role in shaping the respective open source communities. To some extent, however, one would expect that some of the personal motivations that exist in open source software are also applicable to open source production in the life sciences. For example, the indirect financial rewards and ego gratification associated with “signaling incentives” discussed above⁹⁰ are likely to be valid incentives across all technological fields, because they appeal to basic aspects of human nature. However, in some cases, it may be that the motivational factors are valid for different reasons, depending on the technology. For example, the hacker battle cry, “information wants to be free,” is grounded, at least in part, in the altruistic and utilitarian notion that withholding data tends to harm society. In biology-based open source movements, however, there is an additional moral justification for the same statement. Suppose, for example, that a researcher sequences the genome (i.e., determines the genetic blueprint) of a parasite responsible for a virulent tropical disease. One might apply the principle that “information wants to be free,” and argue that the researcher, by patenting the gene sequences, would harm society by impeding the search for a cure to the tropical disease. However, there is also a strong moral argument that the researcher should not be allowed to patent genetic sequences that have been a part of nature for millions of years, and in principle should be available to all. Thus, even though genetic sequences are statutorily patentable subject matter, the researcher may opt for open source distribution instead, in view of this additional moral consideration.

One way to gauge which motivations will likely be important in open source crop engineering and open source drug development is to look at the issues that motivated their founders. From this analysis, it is evident that altruism and a strong sense of social responsibility have played a central role in the genesis of these open source movements. Consider, for example, the following observations published on the BIOS web pages:

Currently, biological technologies are being focused almost exclusively on high-margin applications that do not solve the problems of poor people, nor involve them in creating solutions. Applications to industrial agriculture or “life style” pharmaceuticals are legion, whereas sensible approaches to solving the food production or public health challenges of poor people languish.⁹¹

* * *

⁸⁹ As of this writing, the open source contributions that have been made are primarily those of the founders.

⁹⁰ See *supra* p. 11 and notes 30-32.

⁹¹ BIOS Home, <http://www.bios.net/daisy/bios/home.html>.

It is essential for those who wish to see a change – a real change in social justice and fairplay – to rectify the extraordinary barriers to economically robust innovation presented by our current “innovation” system, including capital, Intellectual Property and other barriers.⁹²

Maurer *et al.* made similar types of statements in their proposal to establish the TDI, when they highlighted the injustices in healthcare resulting from decisions for drug development that were based on purely economic considerations.⁹³ These statements by BIOS and Maurer *et al.* send a strong message to potential contributors, and help to establish a set of values for each respective open source community.

2. *The Importance of Externalizing Costs of Scarce Resources*

In addition to a consideration of a researcher’s personal incentives (i.e., “carrot” incentives), the ability of a researcher to externalize costs of scarce resources is an important part of analysis for determining whether he can contribute to open source production. In some cases, the inability to externalize such costs will be enough to prevent contributions from even the most motivated would-be contributor.

To understand the importance of externalizing costs, one must keep in mind that an open source contributor typically is not paid for his contributions. Accordingly, in most cases, he cannot recover his development costs from downstream users who take advantage of his contribution. In open source software development, this is usually not much of a problem, because the costs associated with developing open source software movement are relatively minimal. Costs are low because (1) computer hardware is abundant and cheap, making entry costs low; (2) software duplication costs are negligible and often are spread out over a large number of users (e.g., through the downloading process); and (3) fixed overhead costs are low. Accordingly, often a programmer can simply absorb the cost of open source software development himself, rather than externalizing it (i.e., shifting the costs to someone else).

In contrast, open source production in certain fields, such as the life sciences, utilizes “scarce” resources, such as reagents, equipment, or laboratory space. While it is possible that a researcher will absorb the costs of open source development through the use of his personal funds, equipment, or supplies, a more likely scenario is that the researcher will have to externalize at least some of the costs. This is especially true if the project requires expensive materials or specialized custom-made equipment. However, if the researcher is unable to externalize these costs, the open source research will not proceed, even if the researcher has strong convictions about the merits of open source production.

A few simple examples illustrate this point. Suppose a biology graduate student wishes to contribute to open source crop development, strongly motivated by one or more of the “carrot” incentives discussed above. Because the student’s advisor

⁹² *Id.*

⁹³ Maurer, *supra* note 14.

controls the equipment and reagents of the laboratory, the graduate student cannot proceed with open source research if prohibited by the advisor. In other words, the graduate student is precluded from contributing to the open source movement because she is unable to externalize her costs of research. In another scenario, a researcher with little discretionary funding may be hard pressed to provide an economic justification to his superiors for using precious lab resources to develop materials for open source movement, even if the researcher belongs to a prestigious industrial or governmental research laboratory. Situations such as these have the negative effect of narrowing the universe of possible participants in the open source movement.

From this analysis, a general rule emerges: the inability of a researcher to externalize costs of open source production involving scarce resources may bar the researcher from contributing, notwithstanding the researcher's enthusiasm for the open source movement. As a corollary, open source production involving non-scarce resources generally will not be limited by the need to externalize costs, but rather only by the extent to which members of the open source community are personally motivated to contribute.

B. "Stick" Incentives Revisited

This section re-examines the "stick" incentives in connection with their use in maintaining and protecting an open source production model in the life sciences.⁹⁴ As in the discussion of the corresponding "carrot" incentives, some of the analysis here is difficult because the examples considered here (i.e., open source crop engineering and drug development) are not yet mature. For example, it is difficult to discuss the influence of norms for enforcing proper behavior within these open source communities, because the communities are just now forming. Thus, the discussion is limited to the fundamental underlying legal structures that may be used to protect an open source movement based in the life sciences. As discussed below, new complex issues will arise with respect to open source licenses, because open source contributions in the life sciences are primarily protected by patents, rather than copyrights.

1. *The Limitations of Patent Protection for Open Source Development*

In some respects, copyright protection is especially complementary to the open source production model. As noted previously, copyright protection attaches immediately to an original work that is "fixed in a tangible medium of expression".⁹⁵ This remarkably simple requirement has the effect of conferring copyright protection to virtually any open source programmer in the U.S., no matter how unsophisticated or economically disadvantaged. For example, the copyright protection afforded to a fourteen-year-old programmer contributing to an open source movement from his bedroom is the same as that afforded to a professional programmer writing programs at a major software development corporation.

Another reason that copyright protection is particularly suitable to open source

⁹⁴ Much of this discussion is also applicable to any open source production model that relies on asserting a patent right to enforce an open source license.

⁹⁵ 17 U.S.C. § 102.

software production is because it automatically extends to any derivative work. This feature facilitates the enforcement of open source licenses by allowing the cost of enforcement to be externalized to the party most capable or willing to pay. For example, if several programmers contribute *seriatim* to an open source program that is subsequently misappropriated, any one of the contributing programmers may bring an action for copyright infringement to enforce the open source license. In practice, however, the programmer who brings the action will likely be the one who has the greatest economic motivations for doing so.

Whereas copyright protection appears to be especially complementary to the open source production model, patent protection appears to be the opposite, for several reasons. First, there are high barriers to obtaining patent protection. For example, patent protection in the U.S. does not attach when the invention is completed, but rather when the U.S. Patent and Trademark Office grants it. The granting of a patent is a long and arduous process, typically requiring an examination period that can be three years or more.⁹⁶ Moreover, during examination of a patent application, an applicant normally must submit detailed technical and legal arguments to rebut an Examiner's allegations of unpatentability (i.e., that the invention is not novel,⁹⁷ that it is obvious in view of prior work,⁹⁸ or that the applicant's disclosure fails to meet certain formal requirements⁹⁹). Because most inventors lack the technical and/or legal background to make such arguments, they must hire specialists (patent agents or patent attorneys) to make those arguments for them.¹⁰⁰ This can be, and often is, quite costly.

The high cost and knowledge barriers associated with obtaining patent protection suggest that "small inventors" (i.e., inventors who are not knowledgeable about patent law and/or who are economically disadvantaged) will often fail to obtain patent protection for their inventions. This, in turn, has profound consequences for the open source production model in technological fields, such as the life sciences, where inventions can only be protected through patenting. A small inventor wishing to contribute to the open source movement in such a field would be faced with two poor choices: (1) disclosing the invention without any intellectual property

⁹⁶ For applications filed on or after June 8, 1995 the patent term is 20 years from the date of filing and begins on the date that the patent issues. An example of this is provided for by the Uruguay Round Agreements Act § 532(a)(1), Pub. L. 103-465, 108 Stat. 4809 (1994). The U.S. Patent and Trademark Office (USPTO) keeps track of any delays that occur during the examination of a patent application and adds time to the patent term if the USPTO itself is responsible for the delay. 35 U.S.C. § 154 (2005). Note that delays up to 14 months before a patent examiner contacts an applicant regarding an application is not considered unreasonable by the USPTO. Thus, pre-examination delays of 14 months or less are not added to the patent term.

⁹⁷ 35 U.S.C. § 102 (2005).

⁹⁸ 35 U.S.C. § 103 (2005).

⁹⁹ 35 U.S.C. § 112 (2005).

¹⁰⁰ A patent agent is a person who does not have a law degree but has passed the Patent Bar Exam that is administered by the U. S. Patent and Trademark Office, whereas a patent attorney is an attorney who has passed the Patent Bar Exam. Historically, relatively few people have taken and passed the Patent Bar Exam. In part, this is because the requirements for being able to sit for the exam tend to exclude many people. Not only is knowledge of patent law required, but also one must provide evidence of competence in some technical field, usually by showing that one has a technical degree or has taken many technical courses. See U. S. PATENT AND TRADEMARK OFFICE, GENERAL REQUIREMENTS BULLETIN, <http://www.uspto.gov/web/offices/dcom/olia/oed/grb17feb05.pdf> (listing details about prerequisites for agent exam).

protection, thereby in effect dedicating it to the public domain; or (2) disclosing the invention to others subject to private contracts with terms similar to those of existing open source licenses.

The first option is objectionable, because a small inventor without any intellectual property protection would be powerless to stop an opportunistic party with economic motivations and greater legal and economic resources from enclosing the invention. While it is true that, strictly speaking, a later inventor cannot obtain a patent under U.S. law based on something that he or she found in the public domain,¹⁰¹ this restriction, in and of itself, probably would not be enough to sustain the open source production model. An economically motivated party may, for example, patent around the publicly disclosed invention (i.e., creating a “patent fence”), thereby thwarting further open source development related to the original invention. Since there is, in this scenario, no licensing agreement between the small inventor and the opportunistic party, and since the small inventor has not obtained any intellectual property rights, the opportunistic party can create such a patent fence without any fear of retaliatory lawsuits from the small inventor. This situation is quite different from the case of copyrightable subject matter, when a copyright holder retains a right to prevent others from creating derivative works based on the copyright holder’s original work.

The second option, namely disclosing the invention to others subject to private contracts with terms similar to those of existing open source licenses, is problematic for two reasons. First, there is the issue of transaction costs: formation of such private contracts must precede the disclosure of the invention to the licensee, because the small inventor has not obtained patent protection.¹⁰² However, if many parties happen to be interested in a particular invention of the small inventor, the costs incurred by the small inventor in dealing with multiple potential licensees may be prohibitive. Thus, transaction costs could limit the proliferation of the invention, even though the small inventor originally intended the invention to be freely available under an open source production model. Another problem with the formation of individual contracts is that, even if a small inventor were to draft a private contract that attempts to achieve the same result as an open source license, there is no guarantee that all potential licensees would agree to the same terms. This is especially true in situations where there may be drastically different levels of bargaining power between the small inventor and the potential licensee. To some extent, this lack of contractual uniformity may be mitigated if the small inventor tries to adopt a licensing agreement from a central organization as a standard non-negotiable license. However, it still remains far from certain that all potential licensees would agree to such a license without further modification.

¹⁰¹ See 35 U.S.C. § 102(a) (preventing this type of behavior by prohibiting anyone from obtaining a patent if the invention was “known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by applicant for patent”).

¹⁰² See Benkler, *supra* note 26, at 445-46 (noting that contracts and property rights lead to higher transaction costs in open source movements for information production).

2. *“The Protected Commons”*: A New Legal Structure for Protecting the Inventions of a Small Inventor

The previous subsection outlined the complex legal problems faced by the small inventor with respect to securing legal protection for his open source invention without obtaining patent rights. One possible way to address many of these problems is to form a confidential “protected commons,” a new legal structure recently proposed by the CAMBIA BIOS initiative.¹⁰³ Within the protected commons, small inventors can publish their inventions or improvements to an audience consisting of parties who have already signed a licensing agreement. This venue is advantageous to the small inventor for many reasons. First, the small inventor would not need to obtain patent protection because the other members are already contractually bound by a licensing agreement. Second, the transaction costs associated with negotiations and the potential problems of contractual non-uniformity are eliminated, because each party desiring to have access to the confidential protected commons must sign a standard licensing agreement. Third, even though the inventions are kept in a confidential protected commons, the requirement that participants sign a licensing agreement in order to obtain access to the protected commons is a minimal burden. Thus, the inventions are essentially still freely accessible, even though they are not immediately transparent to non-signing parties.¹⁰⁴

Note, however, that even though the confidential protected commons offers some protection of the small inventor’s invention, it is not without drawbacks. One potential drawback relates to the difficulties of obtaining a patent when a later inventor using methods from the confidential protected commons to develop proprietary products attempts to patent those products. Under U.S. law, for example, a patent application must contain sufficient disclosure for “any person skilled in the art . . . to make and use [the invention].”¹⁰⁵ Thus, the patent applicant must divulge the methods that he or she used to produce the invention. However, if some of those methods were taken from the confidential protected commons, then the patent applicant would not be able to obtain a patent for his or her proprietary product without violating the confidentiality clause of the licensing agreement.

Another problem with the confidential protected commons relates to the possibility that a third party with economic motives may legally reverse engineer the techniques or products that belong to the commons. Because the unpatented inventions of the commons are, in essence, trade secrets, a third party who discovers these inventions through reverse engineering also has the right to use the invention. Although the third party may not patent the reverse engineered invention,¹⁰⁶ it may

¹⁰³ BioForge, *supra* note 68.

¹⁰⁴ From the vantage point of the small inventor, the confidential protected commons is more attractive than other previously proposed venues for sharing technology, such as a patent pool. In the scenario being described here, small inventors by definition do not have the means for obtaining a patent, so they cannot participate in a patent pool, except as a licensee. See JEANNE CLARK ET AL., U.S. PATENT & TRADEMARK OFFICE, PATENT POOLS: A SOLUTION TO THE PROBLEM OF ACCESS IN BIOTECHNOLOGY PATENTS? (Dec. 5, 2000), <http://www.uspto.gov/web/offices/pac/dapp/opla/patentpool.pdf> (noting that a patent pool refers to an agreement between multiple patent owners to license their patents to each other and/or to third parties).

¹⁰⁵ 35 U.S.C. § 112.

¹⁰⁶ Horne, *supra* note 52.

nevertheless attempt to erect a patent fence around the invention, thereby causing further research in this area by members of the open source community to come to a grinding halt. The third party can do so with impunity because it has not signed an open source licensing agreement. Even worse, there may be situations where a third party would have strong economic motivations to reverse engineer an invention in a confidential protected commons. Suppose that there is a large open source community that has a confidential protected commons where much of the research is based on an original unpatented invention. A third party, if it is able to perceive the existence of the original unpatented invention, may target it for reverse engineering. If successful, it could try to obtain a patent fence that essentially would interfere with the further research by other members of the open source community in this area. Thus, the other members of the open source community would be forced to either (1) abandon research in this area, or (2) take a license with the third party and pay royalties.

In summary, the problems that small inventors face under patent- or contract-based licensing agreements seem to suggest that an open source production model that relies on such agreements would be far less “open” compared to the open source software movement. In the software arena, the copyright protection that attaches when original code is “fixed in a tangible medium” helps to ensure that nobody in the open source software movement misappropriates the work of another (i.e., “good fences make good neighbors”). In technological fields where the intellectual property rights are grounded in patents, however, it appears that generally only those participants who have the economic and legal means to obtain patents for their inventions can safely contribute to an open source movements without fear of misappropriation by another. An exception to this general rule occurs when a confidential protected commons is formed to provide a venue for the disclosures of small inventors. Otherwise, as a practical matter, economic and legal barriers are likely to deter individual small inventors from contributing to such open source movements.

IV. CONCLUSIONS

From the analysis above, it appears that small inventors who cannot obtain patent protection for their invention face many legal problems if they wish to release their invention under an open source production model. These problems arise primarily because an open source license that uses patent rights for enforcement attempts to reconcile two opposing forces: the high cost of obtaining patent protection versus the desire to distribute the technology freely. To some extent, these problems are mitigated by the creation of new legal structures, such as the confidential protected commons discussed above. However, even under such a nuanced system, one can see that confidentiality itself is a double-edged sword; while it protects the inventions of the small inventor, it also interferes with the full disclosure requirement in patent applications, making it difficult for commons members who wish to patent an improvement based on a confidential unpatented invention.

A large part of the discussion surrounding the open source production model has centered on the problems faced by the small inventor seeking legal protection for his open source contribution. One might query whether such open source contributions generally are worth protecting in the first place. The answer, of course, is

affirmative – there are strong public policy reasons for doing so. The first reason relates to the *criticality* of the small inventor's invention. Because a small inventor by definition has limited financial means, he typically will focus his research efforts on fundamental technologies that are critical to him or his local community.¹⁰⁷ Since these fundamental technologies may be critical to other communities as well, it would be sound policy to encourage the small inventor to share the invention by providing effective legal protection for it. The second reason relates to enhancing the *diversity* of innovation. The creation of effective legal protection for the open source contributions of small inventors will encourage more small inventors to participate. Accordingly, the open source community will contain more diverse technologies, which will tend to promote further innovation.

For all of these reasons, it is important to continue developing new legal protections for the inventions of small inventors. While the precise nature of such protections is outside the scope of this paper, such protections should have the following attributes: (1) low cost and simplicity, (2) rapid attachment of an intellectual property right, (3) ease of enforceability, and (4) the option to externalize enforcement costs. As open source production in the life sciences holds great promise for improving the lives of millions of people, the development of such legal protections is an important challenge that must be met.

¹⁰³ See, e.g., BIOS, A Hypothetical Researcher use BioForge, <http://www.bios.net/daisy/bios/221.html> (providing that a small inventor could be focused on a local crop production issue or a local health issue).