ELSEVIER

Research Policy 32 (2003) 1179–1198

research policy

www.elsevier.com/locate/econbase

Guarding the commons: how community managed software projects protect their work☆

Siobhán O'Mahony*

Negotiations, Organization and Markets group, Harvard Business School, Baker Library West 186, Soldiers Field, Boston, MA 02163, USA

Abstract

Theorists often speculate why open source and free software project contributors give their work away. Although contributors make their work publicly available, they do not forfeit their rights to it. Community managed software projects protect their work by using several legal and normative tactics, which should not be conflated with a disregard for or neglect of intellectual property rights. These tactics allow a project's intellectual property to be publicly and freely available and yet, governable. Exploration of this seemingly contradictory state may provide new insight into governance models for the management of digital intellectual property.

Keywords: Software; Public goods; Open source; Intellectual property; Common pool resources

1. Introduction

> It's a kind of cool to bring the power to where it belongs and what's really exciting about working with the hackers is that .... if you're [a] hacker and you leave [a firm] and you've done this work for hire, then you no longer own your product. Well, with open source, you own your stuff right? (*Contributor, GNOME a Graphical User Interface (GUI) Desktop Project.*)

Open source software shares some similarities with privately produced pure public goods, but also differs from traditional definitions of public goods in important ways. It is owned and governed by a bounded community of individuals as opposed to a government, consortium or single private actor. While open source software is publicly available and redistributable, contributors to community managed software projects do maintain and exercise rights over their work.[1] A community managed software project is an open source or free software project initiated and managed by a distributed group of people who do not share a common employer. Project contributors may consider themselves to be associated with either the free software or open source social movements or unaffiliated with a social movement. Contributors may be sponsored by firms, but they are not employees of the project

☆ This research was in part supported by the Stanford University's Center for Work, Technology and Organization, the Stanford Technology Ventures Program, and funds provided by the Alfred P. Sloan Foundation for the Social Science Research Council's Program on the Corporation as a Social Institution.

* Tel.: +1-617-495-0875; fax: +1-617-496-7379.
E-mail address: somahony@hbs.edu (S. O'Mahony).

[1] In suggesting that open source software may differ from pure public goods in important ways, the intent is not to question the degree to which the public benefits from the provision of open source software, but to reconcile assumptions about open source software with empirical findings from the field.

0048-7333/03/$ – see front matter © 2003 Elsevier Science B.V. All rights reserved.
doi:10.1016/S0048-7333(03)00048-9

and project relations are not guided by employment relations.[2]

Observers of the open source phenomena questioning why contributors to community managed projects would give their work away for free have neglected to examine what is given away (code) and what is retained (rights). The property rights programmers exercise to sustain their collective goods are thus not fully appreciated. Empirical examination of how six community managed projects manage their work indicates that they use the legal techniques of copyright, trademark, software licensing, and incorporation to protect their collective works. However, the types of threats that they defend against differ from the threats typically targeted with these legal techniques.

First, I explicate attributes that open source software shares with public good and common pool resource models. Next, I discuss the research methods and mechanisms used to protect six community managed software projects. Analysis of these mechanisms motivates a discussion of the practices used to manage common pool resources. A conceptualization of open source software that more explicitly recognizes its collective governance is advanced. I conclude by exploring ways in which this conception might challenge existing assumptions about the nature of community managed software and inform future research.

2. Public goods and common pool resources

Open source software is often characterized as a privately produced public good (Kollack, 1998, 1999; Lerner and Tirole, 2002a; Johnson, 2001; Bessen, 2001; Weber, 2000; Hars and Ou, 2000). First, it is the product of private-collective efforts. Volunteer contributors, sponsored contributors, firms, governments, and non-profits all may contribute hardware, software, or their expertise to community managed projects. Second, it can be considered a public good because it is non-exclusive and joint in supply. A good that is non-exclusive is available to one if it is available to all. A good that is joint in supply is indivisible: its availability to others does not diminish when consumed by one individual (Snidal, 1979).

Open source software meets the definition of a non-exclusive good, as it is publicly and freely available (although it can also be purchased for a fee by those desiring the convenience of a commercial format).[3] It can be downloaded from the Internet and used freely even by those who did not contribute to its development. Contributors to open source projects do not wish to exclude others.[4] Open source software could also be considered joint in supply as increased use of it does not diminish its value or supply. Olson argued that when the benefits from collective contributions are non-exclusive, rational actors would have inadequate incentives to contribute to such ventures and thus be more likely to under-invest in public goods (Olson, 1965). Yet, in the case of open source software, thousands of volunteers[5] donate their private resources to produce open source software with the knowledge, and even hope, that non-contributors will also benefit from their efforts.

Scholars have addressed this puzzle by identifying benefits that might be exclusive to private contributors but not to potential free riders. Thus, von Hippel and von Krogh consider the open source development model to be a 'private-collective' model of innovation: privately funded with collective benefit (2002, p. 18). They argue that programmers contribute freely to the provision of a public good because they garner private benefits from doing so. For example free riders do not get the private learning benefits that contributors accrue from developing software for open source projects. Nor do freeriders get code developed to satisfy precisely their needs (von Hippel and von Krogh, 2003).

[2] I use the term community managed software project to distinguish from open source and free software projects that can be sponsored and managed by firms, for firms can also start and manage open source projects.

[3] Firms selling products and services that use open source and free software contribute value by packaging it with other software, installing it on hardware and/or providing service and support contracts.

[4] The very definition of open source prohibits discrimination against persons, groups or fields of endeavor (Open Source Initiative, 2001, http://opensource.org/docs/definitions.php).

[5] As of this writing, more than 500,000 software users and/or contributors are registered on http://www.sourceforge.net, and 12,000 users are registered on http://www.savannah.gnu.org, the two largest hosts to free software and open source projects. While some contributors to open source projects may be sponsored by firms, they are still volunteers to community projects.

Lerner and Tirole (2002a) also examined benefits that might be exclusive to contributors of community managed projects. They reasoned that project contributors who signal their talents to the market via their contributions could increase their reputations and gain private marketplace rewards as a result. The career benefits that can be obtained in this way might exceed the benefits that could be obtained by retaining it under proprietary control. This argument was found to be partly correct in a study by Hann et al. (2002). While not all open source volunteer contributors experienced a reported increase in their work wages, those with a higher ranking within a particular open source project did indeed experience higher wages in their regular full time jobs. (However, if those in leadership positions also have greater skill and experience, this effect could be confounded by salary increases earned independent of leadership positions on open source projects.)

Additional empirical work elaborates upon the nature of private benefit obtained by contributors to open source projects. Lee and Cole (2000), Hars and Ou (2000), Lakhani et al. (2002) have all found that programmers contribute to open source software for both internal (altruism, fun, reciprocity) and external (improving job and career prospects) reasons. Lakhani et al. (2002) found that the intrinsic value of participating, contributing, and learning from highly skilled peers in a technical community is important in explaining the attraction of many contributors to open source projects. Butler et al.'s (in press) study of the management and maintenance of on-line groups also showed that social benefits were important to explaining why people contribute to on-line communities. They found that expected benefits from participating in on-line groups predicted the involvement of contributors and, in particular, community list managers.

While this research explains why people might contribute to a privately produced public good, how a 'private-collective' innovation model (von Hippel and von Krogh, 2003) sustains itself remains unclear. As Lerner and Tirole (2002b) point out, the terms "open source" and "free software" refer to the licensing terms associated with a piece of software.[6] Licenses may guide the terms of use, but the possibility of hijacking a community managed project always exists (Lerner and Tirole, 2002b). A project is hijacked when a commercial vendor adds proprietary code to the community's work and attempts to privatize it (Lerner and Tirole, 2002b). If this is true, what prevents such outcomes from happening? How are the commons protected?

If a commercial vendor hijacked a community managed project, the future stream of benefits that would stem from the collective resource would be made unavailable to the community. This type of problem shares some features with non-renewable resources or common pool resource problems. Hardin's (1968) "Tragedy of the Commons" theorized that, if all individuals maximized their gains when drawing from non-renewable resources, these resources would, over time, diminish. Because the individual cost of using the resource would always be less than the collective cost, individuals would have no incentive to show temperance (Hardin, 1968). People following their own short-term interests would produce outcomes that were not in anyone's long-term interests (Ostrom et al., 1999). Ostrom and her colleagues (Ostrom, 1990, 1999; Ostrom et al., 1994, 1999) have theoretically and empirically challenged Hardin's assumptions as well as his dismal conclusions (1968). They argue that his metaphor mischaracterizes the problem of common pool resources.

The tragedy of the commons only becomes a tragedy if the actors using the commons are "norm-free maximizers of immediate gains, who will not cooperate to overcome the common dilemmas they face" (Ostrom, 1999, p. 493). The common resource pool perspective recognizes human actors as capable of cooperating with each other and of establishing norms and social mechanisms to encourage and reinforce cooperative behavior.

Recent reviews of field and experimental studies (Ostrom, 1999; Ostrom et al., 1999; Sneath, 1998; Schlager, 1994) indicate that groups can learn to solve problems and develop solutions to manage common goods in sustainable ways. In some cases, locally designed mechanisms to manage natural resources outperformed more centralized solutions (Schlager, 1994). Thus, and thankfully so, the tragedy of the commons may be overstated.

Ostrom and colleagues (Ostrom, 1990, 1999; Ostrom et al., 1994, 1999) have identified two primary characteristics of such dilemmas: the difficulty

[6] The term open source is also often used to refer to a model for developing software.

of exclusion and subtractability. Like public goods, common pool resources are either non-exclusive or the costs of excluding others is effectively prohibitive. Common pool resources are also subtractable, which is the opposite of joint in supply. Goods that are subtractable are reduced in value with continued use. Exploitation by one reduces the availability of the resource for others (Ostrom et al., 1999).

To return to the comparison between public and private goods, we have established that open source and free software is privately produced and non-exclusive. This meets the definition of both public goods and common pool resources. The next dimension is more ambiguous: is open source software subtractable or joint in supply? If one person downloads software for his or her personal use, the amount available for the next person is unchanged. However, it requires more protections than those offered by the public domain. To remain open and publicly available, it must be protected from proprietary appropriation. Thus, open source and free software appear to be joint in supply, but are in fact vulnerable to usage that would threaten its availability to all. Use of the software will not diminish in the present, but the future stream of benefits is at risk.[7] Thus, it is not subtractable in the manner previously defined, but it does share some features of a common pool resource problem in that the regulation of behavior in a manner that maximizes collective gain is of concern. This research empirically examines how community managed projects protect and sustain their work.

3. Methods

This research was guided by an inductive, qualitative approach using ethnographic methods. A qualitative approach can help explain how theoretical principles are enacted in particular cases (Van Maanen, 1998), in particular, those cases that defy existing categories or theoretical explanations. Furthermore, qualitative methods are most suitable for grounded theory building (Eisenhardt, 1989). Grounded theory building has three distinct features: theoretical sampling, the making of constant comparisons, and the use of a coding paradigm to ensure conceptual development (Strauss, 1987). All three tactics were used in this research.

First, multiple sources of information enabled triangulation and validation of theoretical constructs that could withstand analysis from varying perspectives. Data was collected from three primary sources: (1) observation at project and user group meetings, technical presentations and conferences; (2) informant interviews; and (3) project data archived on the Internet that detailed project interactions and structural developments. Over 100 hours were spent observing and meeting informants at 27 different events (project meetings, user group meetings and conferences) between April 2000 and 2001. Seventy-five semi-structured interviews were conducted with contributors to open source projects. A summary of informant information is provided in Table 1. Most (77%) were conducted in person, although some (23%) were conducted over the phone. Gaining an understanding of membership, sponsorship, decision-making, governance and ownership practices used on open source projects was an important focus of the interviews.

Fifty-six percent of informants were identified through face-to-face means while others were identified directly through on-line documentation of their project or from referrals from other informants. About two-thirds of the respondents could be identified as having a corporate sponsor that allowed them to work on community managed projects as part of their employment. The selection of informants was guided by maximizing the variance in perspective of as many

Table 1
Informant attributes

	Percent
Male	0.96
Female	0.04
Independent[a]	0.37
Corporate sponsored[b]	0.63
Identified through face-to-face contact	0.45
Identified through Internet or e-mail introductions	0.55
Interviewed face-to-face	0.77
Interviewed by phone	0.23

$N = 75$.

[a] Volunteer contributor who contributes to open source projects in his or her own free time.

[b] Individual who contributes to open source projects as part of his or her employment.

[7] I thank von Hippel for helping to clarify this point.

different types of actors as possible. Obtaining the perspective of volunteers, volunteers sponsored by firms and firm and non-profit representatives helped to explicate how different motivations, interests and roles affected community practices.

Second, after an initial set of interviews, theoretical sampling guided the selection of six projects to examine at a greater level of detail, projects that were large, technically mature, attracted commercial attention, and that varied in their relations with firms. More information about the characteristics of each project is provided in Table 2. Project data was collected from on-line archives and included documents such as: mission statements, charters, bylaws, meeting minutes, and mailing list archives. Interviews were transcribed and then coded and analyzed along with other sources of qualitative data using Atlas TI software, a qualitative coding application. Over 54,000 lines of text were coded to identify the practices community managed projects used to manage and protect their work. The variance inherent in different types of actors and their different roles facilitated the process of constant comparison and the development of codes and constructs.

4. Findings: tactics to prevent proprietary appropriation

How do community managed software projects protect against the threat of proprietary appropriation? I identified seven primary tactics: (1) adopt software licenses with distribution terms that restrict proprietary appropriation; (2) encourage compliance with licensing terms through normative and legal sanctions; (3) incorporate to hold assets and protect individual contributors from liability; (4) transfer individual property rights to collectively managed non-profit corporations; (5) trademark the brands and logos designed to represent their work; (6) assign trademarks to a foundation; and (7) actively protect the project's brand. Table 3 shows which tactics are used by the six projects and the types of data used to support these findings. After discussing how each of these tactics are used, I examine how to integrate two seemingly irreconcilable concepts: a good that is freely and publicly available, and yet collectively managed and protected. With a more nuanced understanding of the nature of open source software, comparisons with public goods and common pool resources are re-evaluated.

4.1. Licensing terms that restrict proprietary appropriation

The distribution terms associated with many open source and free software licenses are designed to guard against some of the free rider problems associated with the private provision of public goods. Many open source licenses stem from variations of the first free software license, the GNU[8] General Public License (hereafter GPL). This license was developed when, frustrated by the use of proprietary restrictions associated with Unix, Richard Stallman[9] decided to build a free operating system, the GNU system, in 1984. However, Stallman realized that if his free software was in the public domain, it might not remain freely available.

> But then I thought about the question of whether people would change the software and make proprietary versions of it. And I realized that if they did that, they could defeat the whole point, they could negate the effort. Someone could make an improved proprietary version and it could displace the free version. And as a result, people might be using my code but they would not have the freedom that I hoped they would have and I would not have it either, unless I kept using the inferior free version. *But if nobody joined me, it would not do much good*. And so I decided to look for some way I could stop that from happening. And in discussions with a lawyer I worked out the idea of copyleft (*Interview with Richard Stallman, 20 March 2001, emphasis added*).[10]

Stallman realized that if he put his code in the public domain, it could be appropriated and made proprietary. He also recognized, as the quote indicates, that

[8] GNU is based on the recursive acronym "GNU is Not Unix" (Stallman, 1999).

[9] Stallman went on to found the Free Software Foundation, the organization that remains primarily responsible for interpreting and defending the license.

[10] The names of all informants have been disguised to preserve anonymity except in cases where informants permitted their names to be used.

Table 2
Project attributes

Project names/project attributes	GNU project	Linux kernel	Apache webserver	Debian Linux distribution	GNOME GUI desktop	Linux standards base
Founding mission/goal	To develop a free Unix-like operating system	To rewrite MINIX	To create a commercial grade freely available webserver	To develop a free non-commercial operating system	To build a free and easy to use desktop environment	To develop standards to increase compatibility among Linux distributions
Date project founded	January 1984	Summer 1991	February 1995	August 1993	August 1997	June 1995
Date of first release	Spring 1985	1992	April 1995	January 1994	June 1998	May 1998
Primary license used	GPL	GPL	Apache (BSD type)	GPL	GPL	GPL
Foundation formed	Yes	No	Yes	Yes	Yes	Yes
Type of corporation	Public benefit	–	Public benefit	Public benefit	Public benefit	Mutual benefit
Date incorporated	October 1985	–	June 1999	June 1997	August 2000	May 2000
Non-profit status	501(c)(3)	–	501c(3)	501c(3)	501c(3)	501c(6)
Date awarded	∼1987	–	April 2001	June 1999	Pending	2001
Membership association	No*	–	Yes	Yes	Yes	Yes
Companies as members	No	–	No	No	No	Yes
Board officers	Appointed	–	Elected	Appointed	Elected	Elected

* After this study was completed the FSF began accepting associate members.

Table 3
Tactics to prevent proprietary appropriation

	Tactic	In practice	Projects						
			GNU project	Linux kernel	Apache webserver	Debian Linux	GNOME GUI desktop	Linux standards base	Frequency of use
1	Restrict proprietary usage	"GNOME was GPL'd to enforce, to protect software that gets developed from [getting] proprietary features. The license has to be used strategically." (*Project Founder, GNOME*)	I, D, S	I, D, S	–	I, D	I, D	I, D, S	5/6
2	Legal/ normative sanctions	"If the GPL is violated for FSF copyrighted software, then it is up to us to enforce it." (*Founder, GNU Project*)	I, D, S	I, D, S	I	I, D, S	I	I, D, S	6/6
3	Incorporate	"Somewhere along the line, someone might want to sue somebody. We want to make sure that Debian has an umbrella over it, so that people who are working on this potentially as a hobby are not exposing themselves to risk." (*Founder, Debian Project*)	I, D, S	–	I, D, S	I, D, S	I, D, S	I, D, S	5/6
4	Foundation holds copyrights	"So we took the stance that you should have an organization that owns the code, so the Apache Software Foundation does own the copyright on all the code." (*Founding Member, Apache*)	I, D	–	I, D	I, D[a]	–	I, D	4/6
5	Create logo and file trademark	"When the Linux trademark was under attack in 1997, we garnered the funds to hire an attorney to wrest it away from the guy who was trying to use it as a weapon." (*Executive Director, Linux International*)	I, D, S	I, D, S	I, D, S	I, D, S	I, D, S	I, D, S	6/6
6	Foundation holds trademarks	"The [foundation] shall have the right to inspect any products and any and all materials .... Which use the Trademarks to assess the level of consistency and quality of such use." (*Trademark License Agreement, July 2002, Linux Standards Base*)	I	–	I, D	I, D	I, D	I, D	5/6
7	Actively protect brand	"There have been cases where people have called their consulting company Apache Blah or put Apache on a book or software that made it sound like something that we were doing and we have gone out and essentially defended the brand." (*Founding Member, Apache*)	I, D, S	I	I	I, D	I, D	I, D, S	6/6
Number of tactics used			7/7	4/7	6/7	7/7	6/7	7/7	

I: Supported by interview data; D: supported by project documentation; S: supported by secondary sources; –: not supported.

[a] Debian's Foundation does not encourage contributors to reassign their copyrights to the foundation, but the foundation does hold some copyrights.

if he did not protect free software from appropriation, others would be reluctant to share their code as well.

The GPL is a copyright license designed with unique distribution terms, using principles Stallman termed "copyleft." "Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose: instead of a means of privatizing software, it becomes a means of keeping software free" (Stallman, 1999, p. 59). The GPL permits users to access the source code, modify it and redistribute software (Stallman, 1999; Moglen, 1999). However, Section (2)(b) of the license requires that all modifications or derived works released to the public must also be redistributed under the same terms.

> You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all to third parties under the terms of this License (*Free Software Foundation, Section (2)(b), GNU General Public License, Version 2, June 1991*).

This license uses copyright law to achieve a goal that differs from the ends to which copyright law is traditionally applied (Stallman, 2001). The goal of copyright is to restrict unauthorized use, copying, distributing, modifying, and performing.[11] The goal of copyleft is to allow these same activities, but to restrict proprietary appropriation. Unlike software in the public domain, works derived from software licensed under the GPL cannot be made proprietary. With this self-perpetuating clause, the GPL not only establishes a commons (Moglen, 1999), but a fence that protects that commons.

More specialized licenses,[12] have emerged since the word open source was coined in 1998,[13] but the majority of community managed projects use the GPL. Five of the six community managed projects examined as a part of this study licensed their work under the GPL.[14] Project members reported that they adopted the GPL, explicitly with the protections offered by Section (2)(b) in mind:

> The GPL is a sort of the standard license that people were using at the time. So it was just naturally used. For the most part when you see a project appear in the community it tends to be GPL'd (*Sponsored Contributor, GNOME Project*).

This informant confirms what Lerner and Tirole (2002b) found: of 25,792 projects that they studied from http://www.sourceforge.net, 70% of them used the GPL. All contributors to community managed projects that used the GPL reported being very aware of its terms and conditions. One might question, however, how understanding and compliance could be garnered from users who do not necessarily purchase the software.

Without a transaction,[15] agreement with the GPL is established by making the freedoms to use, copy and modify the software contingent upon agreement with the restrictions associated with it. For example the terms of the license hold that modifying and distributing a GPL program, indicates acceptance of the license (Free Software Foundation, 1991). In effect,

[11] Authors of original works are allowed copyright protection under Title 17 of the United States Code. Section 106 of the 1976 Copyright Act grants owners of copyrights, the rights to authorize others to reproduce works, prepare derivative works, distribute and to perform or display the work publicly. The 1980 Amendments to the 1976 Act extended this law to apply to computer programs (defined as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result"). (17 USC 106 and "Copyright Basics" located at: http://www.loc.gov/copyright/circs/circ1.html.).

[12] As of this writing, there are 30 open source licenses that meet the definition of open source (http://www.opensource.org/licenses/index.html). Many, but not all, of these licenses also meet the definition of free software. For the purposes of this paper, only one license is discussed, the GNU GPL. The more recently approved open source licenses are often based on the GNU GPL, but typically have several modifications most often with regard to restrictions on the licensing and use of derivative works.

[13] The word open source was created in 1998 to create a marketing term for free software that was more hospitable to commercial interests (Raymond, 1999).

[14] One project, that does not use the GNU GPL, the Apache webserver project, uses a different license (the Apache license) that is based on the Berkeley Software Distribution (BSD) license. This license does not prevent proprietary appropriation.

[15] It could be argued that users who do not purchase or transact to use the software would not necessarily be bound to the terms of the license. Legal scholars such as (Lee, 1999) have argued that use and acceptance of the licensing terms may be considered adequate consideration.

this means that the GPL only applies to a small percent of users: those who modify and redistribute code. These users are most likely to have commercial intent.

> Ninety to ninety-five percent of the people that use free software have never agreed to the GPL because they aren't undertaking activities governed by copyright law. They are not copying, they are not modifying, they are not redistributing. They grabbed it once from a Website, put in on their computer, and they are just using [it]. They have no obligations under GPL whatsoever. It's only the small five to ten percent of people who actually undertake those activities that are even bound by the license because they've undertaken things governed by copyright law (*Informant, Non-Profit Foundation*).

The GPL targets a specific audience: those most likely to withhold contributions to the code base. Software licensed under the GPL may thus be less attractive to commercial actors who would like to make proprietary derivative works. While it has been argued that this clause inhibits commercial adoption and innovation of GPL code, it can also be considered a clever way to prevent behavior that might threaten the sustainability of freely available code.

This safeguard ensures that code from potential contributors, who might otherwise worry that the commons to which they contribute will be decimated, will be preserved indefinitely for those who have contributed (as well as those who have not). Code that is appropriated for proprietary purposes could be closed and made unavailable to the commons. In fact, Lee (1999) has argued that if free software lacked intellectual property protections, there would be no legal way to prevent the closing of source code.

> "Because of (2)(b), each contributor to a GPLd project is assured that she, and all other users, will be able to run, modify and redistribute the program indefinitely, that source code will always be available, and that, unlike commercial software, its longevity cannot be limited by the contingencies of the marketplace or the decisions of future developers" (Moglen, 1999).

Informants using the GPL do not use it to exclude others from using their work, but to codify a norm of reciprocity and temperance or, in other words, to prevent code from becoming subtractable in the future. Thus, the problem that the GPL targets is not the inability to exclude non-contributors (a free rider problem), but the inability to prevent the proprietary appropriation of work in the public domain. This tactic would however, mean little if it went unsanctioned.

4.2. Legal and normative sanctions

Informants from community managed projects enforce compliance with their licenses through formal and informal means. The method of enforcement may depend upon whether the violation is a commercial or a community one. A "community violation" or a violation by a long-standing community member that is not a part of commercial work may be handled informally through discussions on project or community wide mailing lists. A community violation might include improper use or application of the license, removal of copyright terms, or improper co-mingling of GPL code with proprietary code. A commercial violation is more problematic as it is more likely to involve distribution of the offending software. A vendor selling free software that does not provide the source code or honor a customer's request for the source code is a common commercial violation. Reports of source code violations often come from customers of such vendors who post to a community list or report the infraction to the copyright holder.

> We had three people writing to us saying, "My company bought this product from them, because we need this [. . .] tool for the work we were doing, and we discovered that it looks like it is based on [your software], and darn it they didn't give us the source code (*Informant, Non-Profit Foundation*).

Legally, copyright holders are the only ones with the right to enforce the licenses associated with their work (US PTO, 2001). However, informants felt that the task of monitoring and identifying license violations was a shared one. The Free Software Foundation (FSF), the institution founded primarily to enforce the GNU GPL, encourages those who suspect license infringement to bring it to the attention of those with the rights to formally take action. ". . . [R]eport it [the license infringement]. First, check the facts as best you can. Then tell the publisher or copyright holder of the specific GPL-covered program." (Free Software Foundation, 2001). The FSF will pursue violations of

Box 1. Example of a reported GPL violation

GPL violation?
By anonymous reader on 14 January 2002 12:13 (#5500).

- http://www.oeone.com does not mention which Linux kernel version is used and definitely does not supply kernel source code.
  - They have a flash demo of the product where the autoupdate section clearly shows tools like "xinetd" "alsa-driver" "readline" and "setup". One or more these packages are bound to be GPL'ed (alsa-driver and readline). "Readline" may well be a product of a GNU project.
  - Since no source code is supplied for so many GPL'ed software tools, specifically kernel and other GNU software tools, while their binaries are being distributed for money, I think this may be a possible case of GPL violation.

(*Note*: from http://newsforge.com/newsforge/02/01/10/1922249.shtml?tid=23.)

work to which they hold copyrights and offer assistance to other copyright holders as requested. One informant reported receiving approximately two to five violation reports a week, although only about one every 2 weeks was found to be a valid violation. Box 1 presents an example of a report of a GPL violation from a community member posted to a public open source community website.

Community managed projects also use formal legal mechanisms to alert violators of a licensed work that are much like those used by firms. Legal counsel may be obtained to draft a "cease and desist" style letter to enforce compliance with the license.

> If you break the license, then you get a letter from the lawyer. It's a standard copyright violation. If you break the license, you have violated the copyright, just as if you had stolen music from Napster or illegal pirated DVDs (*Volunteer Contributor, GNOME Project*).

Formal approaches are however, designed to encourage compliance, not to seek damages. To comply with the license, a violator might have to provide their source code or, in a more complex situation, remove GPL code from a proprietary related code base. A violator with co-mingled code could of course resolve the violation by re-licensing the offending proprietary code under the GPL.

> We never threatened damages. We certainly said, If you don't work with us we are going to have to seek damages [. . .]. What we want you to do is not onerous. We want you to comply with the license, which either means stop what you are doing in violation, or bring your product into compliance, which is usually the thing that they pick. Once you bring the product into compliance we ask that you appoint a GPL compliance officer within your organization who is at a high enough level that when we find a GPL problem in the future we can call them up and have a line of communication (*Informant, Non-Profit Organization*).

Those informants who discussed enforcement emphasized that compliance was the primary remedy sought, but also exercised a little known power articulated in the GPL that grants copyright holders the ability to withhold the freedoms granted in the license upon notice of a violation. Section 4 of the GPL states that "any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License."[16] This clause provides some enforcement capability in that copyright holders can ask violators to stop distribution of software that is not in compliance until the violation is resolved. Once resolved, it is up to the copyright holder to restore the violator's rights.

As of this writing, formal legal enforcement actions have not resulted in any known court cases that would require defending the GPL.[17] Companies that have been notified of a violation may take time to comply, but informants reported that most engage in good faith efforts to come into compliance. For example one company that violated the GPL, but has

[16] http://www.gnu.org/licenses/gpl.txt.

[17] From a legal perspective, the GPL has not been tested in a court of law, although legal analysts (e.g. Lee, 1999; McGowan, 2001) have argued that it would be defendable. A legal counsel to the Free Software Foundation has said that if tried, the GPL would be treated like any other copyright violation and draw on precedents from copyright case law.

since come into compliance, announced to customers on their website that:

> We have temporally closed the services due to the fact that our products were against [the] GPL, however the problem has been settled, with all related parties' support. We deeply regret any inconvenience you have experienced, and hope that you will enjoy our services again (Epson Kowa Corporation).[18]

Informants from Fortune 500 firms reported that their firms were well aware of the need to comply with the licensing terms and norms of the community and that they did make changes when violations were detected.

Informal enforcement of license terms draws upon the normative roots of the license and occurs primarily through on-line public forums. The GPL codifies a strong norm of reciprocity that has long been an important part of the programming culture (Williams, 2002; Levy, 1994). Raymond (1999) has described this culture as a gift culture. However, this conception connotes the idea of transactions from one to another. The norm of reciprocity may be best understood as a subtle pressure to contribute to a collectively managed commons.

> The idea of the GPL is that if you want to include our code in your program, your program must also be free software. *It is supposed to put pressure on you* to release your program in a way that makes it part of our community (*Founder, GNU Project and Free Software Foundation*).

There is no limit on what one can take from the commons, but one is expected at some time to contribute back to the commons to the best of one's abilities.

Legal scholars (McGowan, 2001; Lessig, 1999a,b,c,d,e; Lee, 1999; Gomulkiewicz, 1999; Moglen, 1999) have emphasized that the GPL, while grounded in copyright law, cements the values of the community and can help counteract commercial pressures that might violate community norms. An attorney who advises both corporate and community clients on the GPL stated that:

> I don't want to give the legal system too much credit. It [the GPL] also works because there is a whole community ready to accept it. There is a norm of good behavior (*Independent Private Legal Counsel*).

In the eyes of both legal scholars and informants, the GPL's strength stems not necessarily from its legality, but from the public collective opinion of community members. Informants also stressed that the primary vehicle by which they could enforce their license terms was by identifying and critiquing violations on on-line mailing lists and bulletin boards.

> So far no one has had to litigate [...]. And also a very big thing that I have used, is they fight it out in the court of public opinion (*Former Volunteer, Project Founder, Sponsored Contributor, Debian Linux Distribution Project*).

The court of public opinion is conducted on Internet based on-line bulletin boards or mailing lists. All six projects maintain a number of mailing lists to which contributors, users and other interested parties post to on a frequent basis. Almost all informants also participated in non-project based mailing lists[19] that discussed issues related events in the larger open source and free software communities.

There are a couple of theoretical reasons why discussion in on-line discussion forums might have a strong normative effect on community and corporate contributors. First, postings to project mailing lists are public and not anonymous. Communications between any two members are available to all members. Prior research (Fernandez and McAdam, 1988; McAdam and Paulsen, 1993; Gould, 1993; McAdam, 1996) has shown that individuals contributing to a collective good do so interdependently and are affected by the nature and structure of their network relations. The social presence of others can motivate further contributions, despite free rider concerns (Guttman, 1987; Gould, 1993). Norms of fairness and efficacy can inspire individuals to contribute when other individuals in their network contribute (Gould, 1993). From an economic perspective, it is information, and not norms, that can lead to a positive matching effect

[18] Epson Kowa Corporation, located at: http://www.epkowa.co.jp/english/linux_e/linux.html.

[19] Examples of these forums include: Slashdot, freshmeat, kuro5in, Linuxworld, Linuxtoday, Linux journal.

(Guttman, 1987). If people have reliable information on the contributions of others, than they may be more likely to "match" those contributions and contribute as well. Both sociological and economic explanations suggest that any mechanism that enhances the visibility of both norms and information might have a positive mutually reinforcing effect on encouraging individuals to contribute to collective goods.

Informants thought that on-line public forums were an effective and powerful tool to ensure that others who adopted their work did so without violating community norms and licenses. Formal sanctioning was most evident on the GNU project, which is not surprising given the fact that the GPL was first published for this project. As Table 3 indicates, evidence of informal sanctioning was found on all projects. There is suggestive evidence that firms using open source and free software understood and respected both formal and informal sanctioning tactics.

4.3. Incorporate

Five out of the six projects[20] created a legal entity to hold their intellectual property and protect volunteer contributors from individual liability by incorporating and forming a non-profit foundation.[21] How does incorporation protect a community managed project? A community managed project that is instantiated through mailing lists and web pages does not have any legal rights. It cannot sign contracts or hold property. Informants assumed that intellectual property rights could be better defended with a legal entity supporting their projects. They wanted to establish an institution so that projects could live beyond the efforts of their founders, but most importantly, they wanted to gain the same protections and privileges afforded to corporations. Incorporation allows projects to protect volunteer contributors from individual liability, enter into agreements collectively, and protect their code, trademarks, licenses, and brand. Forming a non-profit was a means to an end, a means to allow project contributors to maintain control over their organization in a manner that could be recognized by other commercial and legal entities (Hansmann, 1996).

The special tax exemptions that the United States government grants to non-profit organizations rest upon the assumption that non-profits behave differently from firms (Powell and Clemens, 1998; Weisbrod, 1998). DiMaggio and Anheier (1990) suggest that non-profit organizations signal trustworthiness, primarily because their non-distribution constraint prescribes distributing net income (Hansmann, 1980) and because they typically pursue goals that differ from firms. Non-profit organizations may be more appealing when certain goods are seen as inappropriate for market exchange or requiring special protection from corruption by the profit motive (DiMaggio and Anheier, 1990, p. 144). Hansmann also predicts that such associative organizations are more likely to appear where consumer control is low cost (Hansmann, 1996). The creation of a non-profit foundation to hold a community managed project's assets would seem to indicate that informants wanted to protect their work in ways that were different from market exchange while garnering the rights associated with incorporation.

4.4. Transfer individual property rights to a foundation

Intellectual property rights are better defined and more defensible when owned by a single legal entity. For example having a single and central copyright holder is easier than negotiating the swift cooperation of hundreds of contributors holding individual copyrights, particularly if electronic signatures are not legally recognized. In addition, informants in leadership positions on several projects were advised by their legal counsel that assigning copyrights to a non-profit organization would help to reinforce institutional as opposed to individual liability.[22]

The reasons informants offered for wanting to transfer copyright assignment to a legal entity were to secure legal protections for their code, formally limit

[20] The Linux kernel project remains unincorporated and also used the least number of legal tactics of the six projects studied.

[21] How these non-profit foundations are designed and governed is discussed in O'Mahony (2002).

[22] Independent legal counsels have advised some community managed projects that the liability protection extended to individuals through incorporation may not protect individuals from potential liability associated with their code if individual copyrights are not assigned to the corporation.

individual liability and improve their ability to pursue enforcement actions.

> One of the original goals of the [foundation] in those days was to be the legal shell for the GNU project. That included collecting these copyright assignments. Now the reason that we did that was because of the issues related to enforcement (*Informant, Free Software Foundation*).

Yet, of the seven tactics identified, this tactic was the least popular among the projects and the most controversial. Contributors on four of the six projects transferred individual copyrights to non-profit foundations in order to better secure and defend the project's rights but no project required such transfer as a condition of contribution.

Only two of the six projects explicitly ask individuals to assign their copyrights to the foundation. The GNU and Apache projects were the most active of the six projects in encouraging contributors to assign their copyrights to a non-profit entity, but even they were reluctant to make this a condition of contribution.

> We took the stance that you should have an organization that owns the code, so the [foundation] does own the copyright on all the code. And we are not as good about this as we should be, but we do try to ask the core developers to sign over copyright assignment to us (*A Founder and Project Leader, Apache Webserver Project*).

Debian does not encourage software authors to assign copyrights to their foundation, but states that their foundation is willing to hold them in some circumstances. The Linux kernel project also does not encourage authors to assign copyrights.

> [Copyright] transfer? No, I think that is amoral. Plus, I hate paperwork. So everybody keeps their own copyright. It makes people comfortable about the fact that there is no single copyright holder. So even if I turn over and I rip off the mask, [he is joking] I could not change the way it is licensed. Well in practice, there are not that many people who have even bothered to put their own copyrights on code. There are a couple hundred or something, so to actually change the license would be practically impossible, which is what makes people feel comfortable with it (*Founder and Project Leader, Linux kernel project*).

In explaining his lack of interest in copyright assignment, this informant also articulates a source of conflict that stems from the use of legal mechanisms to control code by those trying to avoid restrictions over their code.

In the hacker[23] culture, individual merit, autonomy, and problem solving ability are paramount (Williams, 2002; Raymond, 1999; Levy, 1994). Hackers believe that "information sharing is a powerful positive good, and that it is an ethical duty to share their expertise by writing free software and facilitating access to information and computer in resources wherever possible" (Raymond, 1999, p. 236). While all of the legal tactics identified could be considered to be antithetical to the hacker ethic to some degree, the strain between these conflicting worlds was most apparent in copyright assignment practices. One informant's comments when discussing who owns the Apache code reflect the dichotomy that exists between a culture that embraces what might be considered a libertarian ethic with one that relies upon legal mechanisms.

> Legally, the foundation owns the code. However, having said that, the people who write the code own the code [. . .]. If you have ever gotten a piece of code into Apache, you own that code, its yours and you should feel proud of it (*Sponsored Contributor, Apache Project*).

This informant first provides a legal answer, and then modifies that answer to provide an answer that is consistency with the norms of the hacker culture.

One reason why copyright assignment may ignite such disparate approaches and practices is because it infringes on the norms of the hacker culture more so than the other tactics discussed. When an individual assigns their copyright to the foundation, ownership is transferred to a non-profit entity managed by project leaders. This act may constitute a greater compromise of the hacker ethic than some were willing to make, as evidenced by the informant who felt that requiring copyright assignment was amoral. On a different project, when other project members perceived

[23] A hacker is defined as "a person who enjoys exploring the details of programmable systems and how to stretch their capabilities . . . one who programs enthusiastically (even obsessively) or who enjoys programming" (Raymond, 1999, pp. 233–234).

an individual as having too much ownership over the code, the conditions of collective ownership were reinforced.

> And you know this was kind of the shock. He says "this isn't your code, this is our code. If you think it is [your code], I will take your code out of this tree, and put this [other] code back in [. . .]." And somebody else basically said to him, "you cannot take his code out because it isn't your code any more than it isn't his code." It's one of those things, checks and balances (*Sponsored Contributor, Apache Project*).

The idea of collective ownership was unfamiliar to all three parties of the on-line conversation recaptured by the informant above. This discussion demonstrates how individuals might keep each other's interpretation of what collective ownership could mean in practice in check.

An informant from one of the projects that was vigilant about copyright assignment articulated the tension embedded in programmers' use of legal mechanisms to preserve a culture that strives to be free of them.

> So, we are in this odd situation as people who want as few legal restrictions as possible on software, but we are pragmatic enough to realize that without some legal protection we would loose the community we built. [. . .] We didn't want a world with lots and lot and lots of licenses. That wasn't our goal. Our goal was to build freedom in the system we had, this flawed system where for some reason somewhere in the 1960s or 70s there was a decision made that copyright law would cover software. We have to work within the decision that exists and it is the default belief in the industrialized world at least. So, we have to build legal systems within copyright laws to defend freedom (*Informant, Non-Profit Foundation*).

Informants from projects that pursued active copyright assignment policies (the GNU and Apache projects) were also more likely to worry about the existence of work-for-hire agreements among their contributors. Firms that hire programmers typically ask them to sign an "Assignment of Pre-Employment Works" or a "Confidentiality and Invention Assignment Agreement" that transfers ownership of all works created by the employee on company premises and using company resources to the company. The Free Software Foundation asks contributing code authors to not only transfer their copyright but to demonstrate that they are free from work-for-hire agreements and clearly own the code they are contributing (Moglen, 2001). Several informants reported revising their agreements with their employers so that they could contribute code to community managed projects, but it was unclear if and how different projects counseled their contributors on how to modify these agreements.

4.5. Trademark brands and logos designed to represent their work

All six projects have developed distinct brands or logos and have filed trademarks[24] with the US Patent and Trademark Office (US PTO) to protect their brands.

> We could still try to claim trademark while getting the foundation incorporated and put it on the web site (*Board Meeting, 28 November 2000, GNOME GUI Desktop Project*).

A primary motivation for filing a trademark was to ensure that the project would be uniquely distinguished and to prevent others from confusing a projects software with other related work. Some informants recognized that some level of active defense of their mark was needed in order to prevent their project name or symbol from ending up in the public domain. Thus, transferring trademark rights to a foundation was an attractive option for most projects.

4.6. Assign trademarks to the foundation

Five out of the six projects have assigned trademark rights to a non-profit entity or foundation created to support the project. For example the Apache webserver project's trademark agreement states that:

> WHEREAS, the Foundation desires to acquire all right, title and interest, Member has to the Trademarks [. . .] Member by these presents does sell, assign and transfer unto the Foundation, and its successors and assigns, Member's entire right, title,

[24] A trademark is a word, name, symbol or device that is used in trade to indicate the source of the goods and to distinguish them from the goods of others (US PTO, 2001).

and interest (*Apache Trademark Assignment Form, Board Meeting Minutes, 27 April 1999*).

Project members collectively manage the trademark, although the specific governance structures vary by project. For example while members of the Apache Software Foundation manage the trademark for the Apache project, project members, as opposed to foundation board members, are the ones who manage the mark for the Debian project. The Linux kernel project was the only project that has a trademark held by an individual. Day-to-day management of the trademark is delegated to a law firm and trusted advisor.

> Interviewer: Why is it [the Linux trademark] assigned to a person versus Linux International [a non-profit foundation]?
>
> Informant: We were afraid of corporations taking over who might do nasty things. [The project leader] was the only one that was trusted (*Informant, Linux International*).

This may be because the Linux kernel project is not directly supported by a non-profit foundation and has resisted efforts to adopt that type of structure. Without a legal entity, the management of intellectual property rights remains at an individual level.

4.7. Actively protect brand

Evidence of activity to protect the brand and collective reputation of the project was found across all six projects. Why would community managed projects worry about protecting their brand? Project contributors felt that protecting their brand was especially important when their work was repackaged and distributed by firms. While it is legally possible for firms to effectively rebrand the work produced by particular projects with minimal to no recognition, project contributors preferred to establish what marketing researchers might call co-branded relationships with firms. When a product or service is co-branded, two contributing brands to the product or service are recognized (Kotler, 2002).[25] Co-branding was attractive to project contributors for several reasons.

First, contributors wanted to ensure that they received appropriate credit and recognition for their work. Firms selling products and services that incorporate community managed software may be required to incorporate appropriate acknowledgement of the project's contributions and, in some cases, the project's copyrights trademarks.

> The only place you found in [firm name]'s [product], that said the word Debian was on the first page of their manual, we would like to thank Debian for their help. If you did not know what Debian was . . . . Is that a group? A people? A piece of software? [with disgust] That was all they gave us. There was never a Debian logo, there was never a Debian anything (*Sponsored Contributor and former Volunteer Contributor, Debian Linux Distribution Project*).

This informant is concerned not about recognition for himself, but for the project. In a commercial context, what mattered to informants was ensuring that the project, as a whole, received the appropriate recognition. The other insight that is apparent from this quote is that a firm's minimal acknowledgement of a project's efforts was a necessary, but not sufficient, step toward maintaining collegial firm and community relations.

The second reason was more pragmatic. Projects wanted to be recognized, but they did not want their work to be confused with that of the firms distributing their work. Project members wanted to preserve the project's reputation, and did not want problems that might be related to a firm's product to be incorrectly associated with their code.

> They [a firm] made CDs that would not install for some reason. People would write to Debian and it would not be [our code]. So Debian started making official CDs. And I attached it to the Debian trademark, which thank goodness is a real trademark, and said you may call your product official Debian if it is made from our CD masters and if you include the source CD and the binary CD (*Former Volunteer Contributor and Project Leader, Debian Linux Distribution Project*).

As this informant explains, by creating a CD master, the Debian project retains some degree of control over

[25] While co-branding has become popular in recent years, few component manufacturers are successful in establishing an identity separate from the product to which they contribute. Well known successful examples who have proved otherwise include Intel, Nutrasweet and GORE-TEX® (Kotler, 2002, pp. 434–435).

the representation of their work and, in doing so, can protect their collective reputation.

While several theorists have emphasized that individual reputation may be a critical motivator to inspire individuals to contribute to open source projects (Raymond, 1999; Weber, 2000; Lerner and Tirole, 2002b), few have recognized the value of the collective reputation. The use and defense of brands and trademarks by community managed projects suggests two things: It suggests that the collective reputation of the group may matter as much, if not more, to volunteer contributors than individual reputation; secondly, it suggests that open source projects value and protect their identity as a community of developers as much as they do the product of their collective efforts.

Three of the six projects used all seven tactics, and all of the projects used at least four tactics. All projects used legal and normative sanctions, created a logo, filed for a trademark, and actively protected their brand. Projects varied most in their willingness to assign intellectual property rights to a non-profit foundation. What is unique about the seven legal tactics examined in this study is that the rights retained by the licensors (individuals and, in some cases, community managed projects) are not the rights for which these tools are traditionally designed, but those most closely aligned with the licensor's values. These values are less concerned with restricting access to the licensor's work and more concerned with preserving public access. To do so, the rights and privileges associated with these intellectual property mechanisms are unbundled and redistributed. For example the GPL grants the rights to use, modify, distribute, and perform to the licensee. This would seem to redistribute the balance of power from the licensor to the licensee. However, these rights are not unbound. Licensees are not allowed to restrict access to the code or impinge on the freedoms of others to make use of the code. Licensees are granted the right to sell the community's work for a profit, but they must do so in a way that distinguishes the community's work and acknowledges their contribution to a commercial product. Thus, a subtle rebalancing of rights is achieved, by transferring power to the licensee, but by making this power conditional on cooperation with community norms.

5. Discussion

This examination was motivated by the discovery of discrepancies between the way informants conceive of and treat the product of their collective efforts, and the way observers discuss and think about open source software. The projects in this study freely provide their source code to the public. Individuals and firms can download it from the Internet and use it to further their personal or commercial goals. However, these projects also use legal and normative tactics to protect their source code from proprietary appropriation and to protect their collective identity and reputation. This research shows that contributors to community managed projects have interests and rights over their work, and that they are interested in protecting their intellectual property. The assumption that open source contributors give their work away must be modified in order to account for the ways in which community managed projects protect their work.

Moglen (1999), among others (e.g. Tuomi, 2000), have argued that Section (2)(b) of the GNU GPL creates a commons "to which anyone may add but from which no one may subtract." Without the legal tactics identified in this study, open source and free software might be in danger of becoming a subtractable good. While the availability of open source software will not diminish with greater use, those who do not comply with the norms of the community could diminish its future value and its availability to others.

> Interviewer: How critical is the GPL to the moral and ethical foundation of . . . ?
>
> Informant: Well the GPL is what makes free software survive in a copyrighted world (*Open Source Firm Founder and Contributor to several projects*).

There is a threat that requires a defense. To stay open and remain publicly available, open source and free software requires protections from risks inherent with work in the public domain. Open source and free software initially resembles a public good, but also shares some of the risks faced by common pool resources.

How do the mechanisms used by community managed projects compare to those used in prior studies of common pool resource problems? Mechanisms to manage common pool resources usually restrict

access to the resource or create incentives to use the resource with more temperance (Ostrom, 1999). Groups that are better able to identify each other are more likely than groups of strangers to draw on trust, reciprocity, and reputation to develop norms that limit use (Ostrom et al., 1999, p. 279; Ostrom, 1999). Groups that are better able to monitor and coordinate activities are more likely to develop mechanisms that can help them sustainably manage their resources. In general, actors are more likely to develop solutions to common resource pool problems, if they have "some autonomy to make and enforce their own rules and they highly value the future sustainability of the resource" (Ostrom et al., 1999, p. 280).

Contributors to the community managed projects in this study envisioned a long future working with the software to which they contributed. For example the Debian Linux Distribution project has been in operation for over nine years, has withstood six different leaders, and continues to grow with over 1000 registered contributing members. It is also clear from the tactics used, that contributors highly value the results of their efforts. Informants spoke of their contributions as investments in their future tools: they are creating code that they will never have to pay someone to use again. Because of the rights community managed projects exercise, it can be said that contributors pool their efforts to create collectively owned and managed resources. Furthermore, because the context of community is primarily on-line public forums, contributors' efforts are highly visible, which facilitates the communication of norms and information that enables monitoring and co-ordination. These factors suggest that this is an environment that is conducive to developing mechanisms to help manage common pool resource problems.

These principles also point to some fundamental differences between common pool resources and open source software. Unlike common pool resources, open source software is always publicly available. Resources do not need to be redistributed or limited on any scheduled basis. What this analysis has established is that open source and free software is not quite a public good and not quite a common pool resource, at least, in the way these types of goods have been previously defined. The problem with classifying open source software as a pure public good is that this conception glosses over some of the more interesting features of community managed software projects. In neglecting to critically examine how old terms are applied to new phenomena, we risk misunderstanding the very mechanisms that may support its resiliency. Before despairing the introduction of murkier levels of complexity, consider von Hippel and von Krogh's charge that "efforts to offer clean and simple models [of private and public goods] for research have excluded from consideration a very rich and fertile middle ground where incentives for private investment and collective action can coexist and a private-collective innovation model can flourish" (2002, p. 11).

In the case of open source and free software, this rich and fertile middle ground can be further explored by rethinking how the law is used to manage rights to digital intellectual property. von Hippel and von Krogh acknowledge a long understood tension in legal theory: the rights of innovators must be balanced with the rights of the public. Copyright protections are granted to allow investors to earn a return on their investment while also providing an incentive for innovators to disclose their works so that the public can benefit from them. This study suggests that the efficacy of copyright law in balancing public and private interests may be called into question when applications of the law are inverted in order to achieve its founding intent. An alternative interpretation is that, in an era of digital intellectual property, copyright law has untapped elasticity in facilitating new possibilities in unbundling and re-bundling rights and re-balancing public and private interests. If this is the case, it is paradoxical that explorations into the elasticity of copyright law have been most thoroughly explored by those least interested in restricting access to protected works.

Digital intellectual property and the creative application of traditional legal mechanisms enable community managed projects to decouple the ability to govern their work from its circulation and possession. This redistribution of rights has been difficult to conceptualize, as these rights are directed towards goals to which the commercial sector is unaccustomed. However, these mechanisms should not be under valued because of their ideological underpinnings. If we leave the world of political science and turn to computer science, the tactics used by community managed projects

form what Stefik (1997a) might call a trusted system. A trusted system has rules governing the terms, conditions and fees for using digital works. In a trusted system, property rights are respected, but transport rights (rights to copy), rendering rights (rights for playing and printing) and derivative works rights (extracting, editing, and embedding protected works in other works) can be disaggregated and managed under different terms (Stefik, 1997b). Stefik argues that while publishers often think that digital technology automatically transfers more power to users, trusted systems can also be used to shift the balance and put more power in the hands of the publishers.[26] Stefik thinks that one reason why this has not happened is because the social framework to support trusted systems is underdeveloped (Stefik, 1997a). While Stefik argues that technology is what can change the balance between publishers and users, it is perhaps ironic that those, who are among the most sophisticated users of technology, are using a combination of legal and normative sanctions to do just that.

The term digital rights management is often used to reinforce rights that restrict use of protected works. However, with a broader definition of such a term, it could also be argued that this is precisely what the community managed software projects in this study do: manage, exercise and defend rights to digital intellectual property. The ability to manage rights to digital works must not become conflated with a specific aim, for this would narrow the types of possible outcomes, just when we are presented with unparalleled flexibility. This research suggests that this type of flexibility may lend itself to solving some traditional collective action problems and that further research in this area could contribute to new insights in the design of new social arrangements to manage digital intellectual property.

Future research must also devote greater attention to the mechanisms that community managed projects use to govern themselves and manage their work, especially when it is distributed in commercial markets or becomes the basis for de facto standards. This is particularly important for policy makers because as Lessig (1999c) suggests, the ownership of software is likely to effect regulatory actors' stance toward it. When the ownership of code is firmly established, governmental authority is more easily assured. When the ownership of code is not as easily established, then too, the ability of government to regulate it will also be less clear (Lessig, 1999c). The degree to which community managed projects continue to evolve in their governance, and the management of their collective resources, may thus affect how governments conceive of and treat their work.

[26] While Stefik is referring to firms, this could also apply to the case at hand.

Acknowledgements

Special thanks to the editors and reviewers for their helpful comments, to my informants for their time and interest and to Stephen Barley, Robert Sutton, Josh Lerner, Carliss Baldwin, George Baker, Victor Seidel, Fabrizio Ferraro, Mark Mortensen, Michael Schrage, Rachel Campagna and Jason Owen-Smith for their always thoughtful comments on the formation of these ideas and earlier versions of this paper. All errors are my own.

References

Bessen, J., 2001. Open Source Software: Free Provision of a Complex Public Good. Located at: http://opensource.mit.edu/online_papers.php.

Butler, B., Sproull, L., Kiesler, S., Kraut, R., in press. Community effort in on-line groups: Who does the work and why? In: Weisband, S., Atwater, L. (Eds.), Leadership at a Distance.

DiMaggio, P.J., Anheier, H.K., 1990. The sociology of non-profit organizations and sectors. Annual Review of Sociology 16, 137–159.

Eisenhardt, K.M., 1989. Building theories from case study research. Academy of Management Review 14, 532–550.

Fernandez, R.M., McAdam, D., 1988. Social networks and social movements: multiorganizational fields and recruitment to Mississsippi freedom summer. Sociological Forum 3, 441–460.

Free Software Foundation, 1991. GNU General Public License, Version 2.0. Located at http://www.gnu.org/licenses/gpl.txt.

Free Software Foundation, 2001. Frequently Asked Questions About the GNU GPL. Located at: http://www.gnu.org/licenses/gpl-faq.html.

Gomulkiewicz, R.W., 1999. How copyleft uses license rights to succeed in the open source software revolution and the implications for Article 2B. Houston Law Review 36, 179–194.

Gould, R.V., 1993. Collective action and network structure. American Sociological Review 58, 182–196.

Guttman, J.M., 1987. A non-cournot model of voluntary collective action. Economica 54, 1–19.

Hann, I., Roberts, J., Slaughter, S., Fielding, R., 2002. Delayed Returns to Open Source Participation: An Empirical Analysis of the Apache HTTP Server Project. Located at: http://www.idei.asso.fr/Commun/Conferences/Internet/OSS2002/Papiers/Hann.PDF.

Hansmann, H., 1980. The role of non-profit enterprise. Yale Law Journal 89, 835–901.

Hansmann, H., 1996. The Ownership of Enterprise. The Belknap Press of Harvard University Press, Cambridge, MA.

Hardin, G., 1968. The tragedy of the commons. Science 162, 1243–1248.

Hars, A., Ou, S., 2000. Working for free? Motivations of participating in open source projects. In: Proceedings of the 34th Hawaii International Conference on System Sciences, IEEE.

Johnson, J.P., 2001. Economics of Open Source Software. Located at: http://opensource.mit.edu/online_papers.php.

Kollack, P., 1998. Social dilemmas: the anatomy of cooperation. Annual Review of Sociology 24, 183–214.

Kollack, P., 1999. The economies of on-line cooperation: gifts and public goods in cyberspace. In: Smith, M., Kollack, P. (Eds.), Communities in Cyberspace. University of California Press, Los Angeles, CA.

Kotler, P., 2002. Marketing Management. Prentice-Hall, Upper Saddle River, NJ.

Lakhani, K., Wolf, B., Bates, J., DiBona, C., 2002. The Boston Consulting Group Hacker Survey, Release 0.73. Located at: http://www.bcg.com/opensource/BCGHackerSurveyOSCON24July02v073.pdf.

Lee, S.H., 1999. Open Source Software Licensing. Located at: http://eon.law.harvard.edu/openlaw/gpl.pdf.

Lee, G., Cole, R.E., 2000. The Linux Kernel Development as a Model of Knowledge Creation. Haas School of Business Working Paper, #00-1246R. Located at: http://faculty.haas.berkeley.edu/glee/Research_Sample_glee.pdf.

Lerner, J., Tirole, J., 2002a. Some simple economics of open source. Journal of Industrial Economics 52, 197–234. Earlier Versions Distributed as Harvard Business School Working Paper No. 00-068 and NBER Working Paper No. 7600.

Lerner, J., Tirole, J., 2002b. The Scope of Open Source Licensing. Harvard Business School Working Paper. Located at: http://www.people.hbs.edu/jlerner/simple.pdf.

Lessig, L., 1999a. The limits in open code: regulatory standards and the future of the net. Berkeley Technology Law Journal 14, 759–769.

Lessig, L., 1999b. Code and the Commons, Keynote, given at a Conference on Media Convergence. Fordham Law School, NY.

Lessig, L., 1999c. Commentaries, the law of the horse: What cyberlaw might teach. Harvard Law Review 113, 501–546.

Lessig, L., 1999d. Open code and open societies: values of internet governance. Chicago-Kent Law Review 74, 101–116.

Lessig, L., 1999e. Reclaiming a commons, keynote address, "The Berkman Center's Building a Digital Commons." Cambridge, MA.

Levy, S., 1994. Hackers: Heroes of the Computer Revolution. Penguin Books, New York.

McAdam, D., Paulsen, R., 1993. Specifying the relationship between social ties and activism. American Journal of Sociology 99, 640–667.

McGowan, D., 2001. Legal implications of open-source software. University of Illinois Law Review 1, 241–304.

Moglen, E., 1999. Anarchism triumphant: Free Software and the Death of Copyright. Saggi, Conferenze E Seminari (38), Roma, also located at: http://emoglen.law.columbia.edu/my_pubs/anarchism.html.

Moglen, E., 2001. Free Software Matters: The Public's Business. Located at: http://emoglen.law.columbia.edu.

Olson, M., 1965. The Logic of Collective Action. Schocken, New York.

O'Mahony, S., 2002. Community Managed Software Projects: The Emergence of a New Commercial Actor. Unpublished Ph.D. Dissertation. Stanford University, 2002.

Open Source Initiative, 2001. The Open Source Definition. Located at: http://www.opensource.org/docs/definition_plain.html.

Ostrom, E., 1990. Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge University Press, New York.

Ostrom, E., 1999. Coping with tragedies of the commons. American Review of Political Science 2, 493–535.

Ostrom, E., Burger, J., Field, C.B., Norgaard, R.B., Policansky, D., 1999. Revisiting the commons: local lessons, global challenges. Science 284, 278–282.

Ostrom, E., Gardner, R., Walker, J.M., 1994. Rules, Games, and Common-Pool Resources. University of Michigan Press, Ann Arbor, MI.

Powell, W.W., Clemens, E.S., 1998. Private Action and Public Good. Yale University Press, New Haven, CT.

Raymond, E.S., 1999. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly & Associates, Sebastopol, CA.

Schlager, E., 1994. Fishers' institutional responses to common-pool resource dilemmas. In: Ostrom, E., Gardner, R., Walker, J.M. (Eds.), Rules, Games, and Common-Pool Resources. University of Michigan Press, Ann Arbor, MI.

Sneath, D., 1998. State policy and pasture degradation in inner Asia. Science 281, 1147–1148.

Snidal, D., 1979. Public goods, property rights, and political organizations. International Studies Quarterly 23, 532–566.

Stallman, R., 1999. The GNU operating system and the free software movement. In: DiBona, C., Ockman, S., Stone, M. (Eds.), Open Sources. O'Reilly, Sebastopol, CA.

Stallman, R., 2001. Frequently Asked Questions About the GNU GPL. Located at: http://www.gnu.org/licenses/gpl-faq.html.

Stefik, M., 1997a. Shifting the possible: How trusted systems and digital property rights challenge us to rethink digital publishing. Berkeley Technology Law Journal 12, 1.

Stefik, M., 1997b. Trusted Systems. Scientific American, March, pp. 78–81.

Strauss, A.L., 1987. Qualitative Analysis for Social Scientists. Cambridge University Press, New York.

Tuomi, I., 2000. Internet, Innovation, and Open Source: Actors in the Network, SITRA. The Finnish National Fund for Research and Development Working Paper.

US Patent and Trademark Office, 2001. What Are Patents, Trademarks, Service Marks, and Copyrights? Located at: http://www.uspto.gov/web/offices/pac/doc/general/whatis.htm.

Van Maanen, J., 1998. Qualitative Studies of Organizations. Sage Publications, Thousand Oaks.

von Hippel, E., von Krogh, 2003. The private-collective innovation model in Open Source Software development: issues for organization science. Organization Science, in press.

Weber, S., 2000. The Political Economy of Open Source Software. BRIE Working Paper 140, Economy Project Working Paper 15.

Weisbrod, B.A., 1998. Institutional form and organizational behavior. In: Powell, W., Clemens, E. (Eds.), Private Action and the Public Good. Yale University Press, New Haven, CT.

Williams, S., 2002. Free as in Freedom: Richard Stallman's Crusade for Free Software. O'Reilly & Associates, Sebastopol, CA.