The Scope of Open Source Licensing

Josh Lerner
Harvard University and NBER

Jean Tirole
University of Toulouse and MIT

This article is an initial exploration of the determinants of open source license choice. It first highlights how the decision is shaped not just by the preferences of the licensor itself, but also by that of the community of developers. The article then presents an empirical analysis of the determinants of license choice using the SourceForge database, a compilation of nearly 40,000 open source projects. Projects geared toward end-users tend to have restrictive licenses, while those oriented toward developers are less likely to do so. Projects that are designed to run on commercial operating systems and whose primary language is English are less likely to have restrictive licenses. Projects that are likely to be attractive to consumers—such as games—and software developed in a corporate setting are more likely to have restrictive licenses. Projects with unrestricted licenses attract more contributors. These findings are broadly consistent with theoretical predictions.

1. Introduction

An extensive body of work has examined the economics of technology licensing. In particular, theoretical studies have intensely scrutinized several aspects of how profit-maximizing firms should license their intellectual property, including the timing of the licensing transaction (i.e., whether before or after the discovery has been made), whether exclusive licenses should be employed,

We thank Larry Augustin, Jeff Bates, and Pat McGovern for access to the SourceForge database. Comments by Ian Ayers (the editor), Yochai Benkler, Keith Bostic, Judy Chevalier, Peter Childers, Jacques Crémer, David Genesove, Brian Kahin, Marten Miklos, Siobhan O'Mahony, Bruce Perens, Bernie Reddy, Larry Rosen, Marcin Strojwas, two anonymous referees, and participants at the conference "Open Source Software: Economics, Law and Policy" at the University of Toulouse, the American Economics Association meetings, and a seminar at Harvard Law School were very helpful. We thank James Hunter, Nicolas Lambert, Bernie Reddy, and Brendan Reddy for their many contributions to the research project. Harvard Business School's Division of Research provided financial assistance. The Institut D'Economie Industrielle receives research grants from a number of corporate sponsors, including France Telecom and Microsoft Corporation. All errors are our own.

The Journal of Law, Economics, & Organization, Vol. 21, No. 1,
doi:10.1093/jleo/ewi002

and the nature of the fees that should be charged (e.g., the trade-off between royalties and flat fees).[1]

But the question of the optimal *scope* of technology licenses has been much less thoroughly scrutinized. More concretely, should the licensee be free to use the technology as he sees fit, being able to commercialize follow-on inventions, or should his use be narrowly circumscribed? This article examines this question in a special context: the licensing of open source software.

The open source process—a method of software development in which contributors freely submit code to a project leader, who in turn makes the improved code widely available—is an interesting arena to start thinking about license scope because the standard considerations (e.g., timing, exclusivity, fee structure) are irrelevant. Users of open source software must typically consent to a licensing arrangement, which may impose a variety of restrictions. For instance, the user may be limited in his ability to distribute a modified version of the program as a proprietary commercial product without releasing the underlying source code.[2]

This article first explores the various considerations that figure into the licensor's decision of how restrictive a license to employ. It highlights the complex set of motivations that may drive the choice of license. It then suggests that permissive licenses will be more common in cases where projects have strong appeal to the community of open source contributors—for instance, when contributors stand to benefit considerably from signaling incentives or when the licensors are well trusted—and restrictive ones commonplace when the appeal is more fragile.

The article then presents an empirical analysis of the prevalence of different types of open source licenses. The analysis employs the SourceForge database, a compilation of nearly 40,000 open source projects that has hitherto been largely unexplored by academics. We focus on two critical characteristics of these licenses:

- Whether the license requires that when modified versions of the program are distributed, the source code must be made generally available. Such a provision is sometimes referred to as a "copyleft" provision. In the empirical analysis in this article, we term such licenses as "restrictive."
- Whether the license restricts modified versions of the program from mingling their source code with other software that does not employ such a license. Such a clause is sometimes termed a "reciprocal" or a "viral" provision. For purposes of the empirical analysis in this article, we term this a "highly restrictive" requirement.

1. Katz and Shapiro (1986) and Gallini and Wright (1990) are illustrative of this literature. Also relevant are those works that explore the real consequences of the licensing decision, whether the impact of this choice on subsequent innovations by the original innovator (Gandal and Rockett, 1995), the decision of rivals to enter the market (Gallini, 1984; Rockett, 1990), or the nature of the competitive dynamics in the industry (Shepard, 1987).

2. Of course, the fact that timing, royalty rates, and exclusivity are not important in this setting means that our ability to draw lessons for the commercial world may be limited.

These licenses, it should be acknowledged, are complex legal documents that have not yet been tested in court.[3] Significant ambiguities remain about their interpretation. What is critical for our analysis, however, is the relative ordering of the restrictiveness of the agreements, not their absolute restrictiveness. We will consider three classes of licenses: unrestrictive [e.g., the Berkeley Software Definition (BSD) license], restrictive [e.g., lesser general public license (LGPL)], and highly restrictive [general public license (GPL)]. (See below for a more complete discussion of these licenses.)

The results are largely consistent with the framework above: more restrictive licenses are more common in projects geared toward end users, with applications such as games and desktop applications, in languages other than English, and designed for a noncommercial user environment or operating system. We explore the robustness of the results to the use of a variety of definitions of the independent variables, as well as to using only the earliest and latest projects added to the SourceForge database. In an exploratory analysis using a much smaller sample, we examine the licensing terms of projects that are spun out of corporations. The results generally correspond with theoretical suggestions. Finally, we show that, consistent with theory, community contributions are greater when restrictive licenses are employed.

2. The Legal Foundations of Open Source Licensing

Software developers have long been able to obtain copyright protection for their works. When for-profit companies manufacture proprietary software products, these copyrighted works are typically licensed rather than sold. By licensing the software, software manufacturers can limit their liability if the product does not work effectively and restrict the rights that the users would normally have (e.g., the ability to simultaneously run the software on several computers). [For a detailed rationale for this approach, see Neukom and Gomulkiewicz (1993).]

In the early days of the computer software industry, however, much of the software was made available without an explicit license governing its use.[4] [For a history of the open source movement, see Lerner and Tirole (2002) and the references cited therein.] By the early 1980s, programmers had become disturbed by instances of behavior that they deemed to be unethical.[5]

In response to these events, MIT programmer Richard Stallman developed a new approach to distributing software in the mid-1980s. Rather than dedicating the software to the public domain, he required users to license the code under

3. One decision that touched on, but did not resolve, these questions was *Progress Software Corp. v. MySQL AB*, 195 F.Supp.2d 328 (D. Mass 2002).

4. Subsequently software was also made available under formal contracts between developers and users. Later (in the personal computer era), software was protected through mass-market “shrink-wrap” licenses.

5. In some instances, firms had solicited contributions from third parties and then sought to enforce intellectual property rights on the software that resulted. In other cases, individuals added a modest amount of new code to software that was distributed without restrictions, which they then sold as a copyrighted proprietary product.

the GNU public license.[6] This license essentially required that the program's source code (the underlying programming commands) be freely available and that modifications to the code must be allowed. One of Stallman's major concerns, however, related to those who sought to commercialize modifications to the code. He limited the ability of software developers to undertake such activities in two critical ways: by ensuring that any derivative works remain subject to the same license and by prohibiting the mixing of open and closed source software in any distributed works. In this way, he limited the danger of commercial exploitation of these discoveries. A variant of the GPL, known as the lesser GPL (LGPL) allows greater flexibility in regard to the "mixing" requirement: in particular, programs are allowed to link with (or employ) other programs or routines that are not themselves available under an open source license. In other respects, though, the LGPL is similar to the GPL.

Meanwhile, several alternative licenses were introduced:

- Perl, a UNIX-based programming language that allows for the automation of many system administration tasks, was originally made available by its founder, Larry Wall, under the GPL. He soon decided that the terms were too restrictive and developed what was termed the "artistic license." With a few limitations, users were free to develop commercial products based on the Perl code. Nor were any limitations placed on the mingling of proprietary and open source code.
- Another variant was the family of BSD-type licenses, which also allowed a great deal of flexibility to users, as long as credit was given to the University of California for the underlying code in the documentation of any derivative version.[7] BSD-type licenses, which have been adopted by many projects (including the Apache Web server), are today the most popular alternative license to the GPL and LGPL.
- Another family of alternative licenses is those introduced by commercial companies that have "opened up" some of the proprietary code (i.e., made the source code available to open source programmers). These programs have frequently added specialized provisions to address copyright and liability concerns of the corporate parent.

In 1998 a variety of open source leaders came together to establish a consistent set of criteria for what constituted an open source license, which they termed the "open source definition." Among the requirements for the license of a program to be considered "open source" were that

- The source code for the program must be available at little or no charge.
- Redistribution of the program, in source code or other form, must be allowed without a fee.

6. GNU was the name of the project to develop a new operating system that Stallman had launched. The license was later renamed the general public license. For a detailed history, see http://www.free-soft.org/gpl_history/ (accessed September 17, 2002).

7. The credit provision was dropped in later versions of the license.

- Distributions of modified software must be allowed without discrimination.
- The distributions of those modifications on the same terms as the original program must be permitted.

This definition was broad enough to both encompass the GPL and those licenses that allow users greater liberty in how they use the code.[8]

Table 1 summarizes the leading open source licenses. For each license that has been approved as falling under the "open source definition" (as well as two other broad classes of related licenses), we report, as discussed in the introduction, whether the license has what we term "restrictive" and "highly restrictive" features.[9]

Despite uncertainties surrounding the enforceability of open source licenses,[10] it is clear that software developers care critically about the choice of license used. Decisions to switch between license types[11]—for instance, the WINE project's recent move from the BSD-like X11 license to the LGPL license[12]—have proven intensely controversial.

3. The Choice of License: A Simple Model

We begin with a highly stylized model of license choice, which we hope will capture the key rationales behind the choice of license.[13] Suppose that an entity,

8. For detailed analyses of the open source definition, see Lee (1999) and Perens (1999).

9. In some cases, those who redistribute the original code must make it freely available, but modifications need not be (e.g., the artistic license). These cases are coded as not being restrictive.

10. The extent to which these licenses can be enforced remains untested in a court of law. These issues are discussed, for instance, in Dodd and Martin (2000) and McGowan (2001).

11. The alteration of open source licenses by project leaders poses several interesting issues. Open source licenses differ somewhat from traditional licenses, such as the "shrink-wrap" agreements that govern the relationship between manufacturers and users of commercial software products, which require the consent of both parties to be effective. Rather, an open source license is best seen as a conditioned permission to use one's property, akin to a landowner who allows hikers to use a path that passes through his property. The project leaders can unilaterally change such permissions, just as the landowner can fence his property without consulting the hikers (or conversely, add to the network of trails). Thus the leaders of a BSD license project would be free to switch to a GPL, and vice versa. Two complications, however, should be noted. First, it is unlikely that open source project leaders can force existing licensees to honor alterations to the terms of licenses. Thus, if a program had been made available under a BSD license and a firm has incorporated the code into a commercial product, the project leaders in all probability cannot subsequently force the firm to make the product available under the GPL. A second complication is introduced by projects where contributors do not assign the copyright for their holdings to a central entity (as the Free Software Foundation and many other sponsors of open source projects require). In these cases, such as the Linux kernel development project, the copyright is in the hands of literally thousands of individual contributors. Any change to the license would probably require the assent of each copyright holder: any holdout could block the shift, unless his software contribution could be rewritten.

12. See, for instance, http://kt.zork.net/wine/wn20020308_117.html (accessed September 17, 2002).

13. In this section we take an optimizing approach. To be sure, the choice of a license may be affected by considerations that either lie outside the standard utility-maximizing paradigms, or may be distorted by a misunderstanding of the implications of the alternative licenses in the choice set. An example of the former is the influence of ideological views: to cite one example, the belief that "software should be free" is sometimes invoked in favor of the GPL. This belief could conceivably be rationalized through its adherents' confidence that future versions of the software will remain communal property. For some adherents, though, this belief is simply a matter of principle.

Table 1. Open Source Software Licenses

License name	Restrictive?	Highly restrictive?	Observations in sample	Observations with activity data
OSI approved licenses				
Apache Software L	N	N	301	121
Apple Public Source L 1.2	Y	N	15	3
Artistic L	N	N	736	223
BSD L	N	N	1,708	618
Common PL	Y	N	34	18
Eiffel Forum L	Y	N	5	3
General PL	Y	Y	18,133	5801
IBM PL 1.0	Y	N	33	7
Intel OSL	N	N	10	6
Jabber OSL	Y	N	20	7
Lesser General PL	Y	N	2,501	1047
MIT L	N	N	395	151
MITRE Collaborative Virtual Workspace L[a]	Y	Y/N	5	1
Motosoto L	Y	N	0	0
Mozilla PL 1.0	Y	N	229	76
Mozilla PL 1.1	Y	N	134	62
Nethack PL	Y	N	16	6
Nokia OSL	Y	N	5	2
Open Group Test Suite L	N	N	1	0
Python (CNRI) L	N	N	162	53
Python Software Foundation L	N	N	0	0
Qt PL	Y	N	136	39
Ricoh Source Code L	Y	N	5	3
Sleepycat L	Y	N	5	2
Sun Industry Standards Source L[b]	N	N	26	9
Sun PL	Y	N	0	0
University of Illinois/NCSA OSL	N	N	1	1
Vovida Software L 1.0	N	N	1	0
W3C L	N	N	0	0
X.Net L	N	N	0	0
Zope PL 2.0	N	N	125	47
zlib/libpng L	N	N	0	0
Other/proprietary	?	?	531	220
Public domain	N	N	820	244

The table summarizes all open source initiative-approved licenses, as well as selected others. The final two columns indicate the number of observations of each license type in the SourceForge database.

Definitions: Restrictive: Y implies that the source code from modifications to the program must be made available. Highly restrictive: Y implies that the program cannot be compiled with proprietary programs.

L, license; OS, open Source; PL, public license.

[a]Licensees can choose between two possible options.

[b]Deviations from certain industry standards, however, must be documented.

whom we will term the "licensor," is deciding (a) whether to make some software available under an open source license and (b), if so, what type of license to employ. The licensor may be a single developer, a group of developers with similar needs, or a corporation. We depict the interactions between the licensor and the community of programmers who are asked to work on their project. The programmers' benefits from working on the project may depend on the choice of license. The licensor must assess how his mixture of motivations, together with project characteristics—such as the environment, the nature of the project, and the intended audience—impacts the project's likely success.

Both the licensor and the open source community may derive commercial and noncommercial benefits from the open source project:

- *Noncommercial benefits* encompass peer recognition, the career advancement associated with involvement in an open source project, the pleasure of solving open source programming problems, and the benefits of being able to tailor the code to one's own specific needs. (This latter benefit exists only for the community, since the licensor will be able to do so whether the project is taken open source or not.) These benefits are denoted $a + c$ for the licensor, and $a + b + c$ for the community.

We define a as a known exogenous parameter that indexes the attractiveness of the project. The variable c is known and increases with the restrictiveness of the license. Our rationale for the term c is that software made available under unrestrictive licenses are particularly prone to "hijacking" by commercial software vendors: in other words, the commercial firm may add some proprietary code to the open source software and take the whole private. (The original project will not be privatized, but there is a risk that the proprietary derivative work will confuse, and perhaps dominate, the market.) While such hijacking need not be socially detrimental—it may take the project to its next logical step or revive interest in an otherwise faltering technology—the action deprives the open source contributors of some of the benefits from the project.[14] This prospect may discourage potential contributors in the first place.[15] Although we do not require that the community's overall welfare increases with c, the interesting case is where it does, as discussed below. Hence it may be useful to think of c as a "concession"

14. For example, they may have to pay for the final software and be unable to tailor it for their own needs. One reason for this fear is that contributors to open source projects enjoy dynamic network effects—see Lerner and Tirole (2002)—and that these network effects may be reduced by competition from a proprietary variant. This argument for restrictive licenses could be rephrased to say that community members make project-specific investments. Hijacking poses the possibility that the members may be "held up": for instance, they may lose the ability to shape the project to meet their particular needs and their contributions become less visible because the open source community loses interest in the project. Several covenants in the restrictive licenses (including that about patent licensing discussed below) can be seen as a Williamsonian (1975, 1985) contractual response to address the danger of such a "hold up" problem.

15. Another possibility that may discourage contributions would be forking, which refers to an internal threat of competing groups moving in different directions and producing incompatible versions of the same initial open source project.

to the open source community. Finally, to capture the uncertainty that the licensor faces regarding the community's enthusiasm regarding the project, we introduce a random parameter b. This community-specific parameter of attractiveness reflects the community's assessment of the prospects for the future development of this project and the uncertainty surrounding this assessment. Known differences between the licensor and the community (such as the benefits from tailoring the code) can be subsumed in the distribution of b.

- *Commercial incentives* are driven by the money that can potentially be made on products and services that are complementary to the open source project. These can be new products and services that emerge concurrently with the success of the open source program and, for a corporate licensor, the increased sales of existing complementary software products. (The firm may also benefit indirectly, such as through an increase in competitive pressure on a rival's proprietary software program that is competing with the open source project.) Let $\pi(c)$ denote this expected payoff. It is natural to assume that $\pi'(c) < 0$: a more restrictive license reduces the opportunities for making money on complementary products, and that $\pi'' \leq 0$.[16]

The *licensor's preferences* regarding license restrictiveness, provided that he opts for an open source license in the first place, are given by $a + c + \beta\pi(c)$, where β represents the weight that the licensor places on the commercial benefits. As described below, we subsume the community's intensity of participation in the project into the probability $p(c)$ that the community actually is willing to contribute to the project. Let $\overline{V}$ denote what the licensor could get by keeping the project proprietary. The licensor chooses an open source approach if and only if

$$\max_{\{c\}} p(c)[a + c + \beta\pi(c)] \geq \overline{V}.$$

The alternative payoff, $\overline{V}$, includes the payoffs on the project itself if it is kept proprietary and, as in the case of the open source option, the potential profit from the sale of complementary products and services. $\overline{V}$ is, of course, independent of c. But as we will discuss further below, $\overline{V}$ may be positively correlated with a: a software program that is dominated by rival software may have a low $\overline{V}$, but is also likely to be unattractive to the open source community, as the programmers will be reluctant to contribute to a project that is unlikely to succeed. In other words, projects that lag behind other programs will be more likely to be turned into open source projects, all else being equal, but because of the uphill battle they face, may be not be embraced by the open source community.

We now turn to the *community's preferences*. Letting α denote the community's relative weight on the commercial benefits from the project, the community's return from participating in the project is $a + b + c + \alpha\pi(c)$.

16. This condition will hold, for instance, if π is linear in c. The condition also ensures that the maximand below is concave.

We make two sets of assumptions. First, the uncertain variable b, denoting the project's appeal to the community, is distributed with a cumulative distribution $F(b)$ and a density $f(b)$ with wide support. This distribution, as is the case with almost all familiar distributions, has a monotone hazard rate, that is, $f(b)/(1-F(b))$ is increasing in b. Second, we assume that the licensor puts at least as much weight on commercial objectives (relative to noncommercial ones) as the community: $\beta \geq \alpha$. Our rationale is that commercial benefits are likely to flow disproportionately to the project leaders: as Lerner and Tirole (2002) document, there are numerous examples where project leaders have parlayed participation in these projects into commercial opportunities. Note that this assumption says nothing about *absolute* preferences. The leadership and the community may both prefer the restrictive license or both prefer the permissive license.

Let $\overline{U}$denote the reservation utility or the opportunity cost for the community's members to participate in the open source project. The community then participates if and only if

$$a + b + c + \alpha\pi(c) \geq \overline{U}.$$

Thus the probability that the community embraces the open source project is

$$p(c) = 1 - F(\overline{U} - [a + c + \alpha\pi(c)]).$$

Let us now consider the decision that the licensor faces. He chooses the open source option if and only if

$$\max_{\{c\}} \{[1 - F(\overline{U} - (a + c + \alpha\pi(c)))]\,[a + c + \beta\pi(c)]\} \geq \overline{V},$$

where the maximization is over some interval $[\underline{c}, \bar{c}]$. ($\underline{c}$ corresponds to the least restrictive license, and $\bar{c}$ to the most restrictive.) Suppose, first, that the community prefers an unrestrictive license: $1 + \alpha\pi'\ \underline{c} < 0$. Because $\beta \geq \alpha$, so does the licensor *a fortiori*. In this case, $c = \underline{c}$, and the choice of license, being a corner solution, is locally independent of the parameters. We now turn to the more interesting case in which $1 + \alpha\pi' > 0$, where the community prefers a more restrictive license. Let

$$\hat{a} \equiv a + c + \alpha\pi(c), \text{ with} \frac{\partial\hat{a}}{\partial c} \in (0{,}1)$$

in the relevant range.

The identity defines a function, $c(\hat{a})$, the degree of license restrictiveness needed to reach an overall level of attractiveness $\hat{a}$. Now define

$$\hat{\pi}(\hat{a}) \equiv (\beta - \alpha)\pi(c(\hat{a})).$$

We can thus rewrite the licensor's objective function as

$$V = [1 - F(\overline{U} - \hat{a})]\,[\hat{a} + \hat{\pi}(\hat{a})]$$

with $(\hat{a} + \hat{\pi}(\hat{a}))' = (1 + \beta\pi')/(1 + \alpha\pi')$. In this setting, choosing the degree of license restrictiveness c amounts to choosing $\hat{a}$. Taking logarithms and differentiating, we obtain

$$\frac{\partial(\log V)}{\partial \hat{a}} = \frac{f(\overline{U} - \hat{a})}{1 - F(\overline{U} - \hat{a})} + \frac{1 + \hat{\pi}'(\hat{a})}{\hat{a} + \hat{\pi}(\hat{a})}.$$

If $1 + \hat{\pi}' > 0$, that is, if $1 + \beta\pi' > 0$, then the licensor prefers a restrictive license, and so $c = \bar{c}$. This corner solution is again locally invariant to changes in the parameters.

We consider next the case of an interior solution, where

$$\frac{\partial(\log V)}{\partial \hat{a}} = 0.$$

We first consider the comparative statics with respect to $\overline{U}$. Differentiating the expression yields

$$\frac{\partial^2(\log V)}{\partial \overline{U} \partial \hat{a}} = \left(\frac{f}{1 - F}\right)' > 0.$$

The overall attractiveness $\hat{a}$, and hence the restrictiveness of the license, must increase with the community's outside opportunity $\overline{U}$.

Because $\hat{\pi}$ is weakly concave, we can derive an additional result. Differentiating the first-order condition shows that

$$0 < \frac{\partial \hat{a}}{\partial \overline{U}} < 1.$$

This relationship implies that participation, as measured by the probability that the community embraces the project, decreases as $\overline{U}$ increases.

We can now derive some additional implications from this analysis. More generally, $\overline{U}$ can be seen as standing for the *relative* attractiveness of alternative projects.[17] Thus we can conclude that the license becomes less restrictive (c decreases) and participation increases ($1 - F$ increases) when a project becomes more attractive to the community, keeping attractiveness for the licensor constant. Formally this will hold when the distribution of b shifts. Let θ represent a known shifter of the project's attractiveness to the community only (e.g., a change in the ability to tailor the code). The distribution of b is $F(b - \theta)$, such that $f(b - \theta)/[1 - F(b - \theta)]$ decreases with θ from the monotone hazard rate property. A project that is more attractive to the community is one with a higher θ.[18]

Examples include

- Projects where the community has more trust in the licensor, which may be the case, for instance, where the licensor is an individual rather than a corporation.

17. One possibility is that the proliferation of open source projects in recent years has led to more competition for open source programmers' attention, and hence an increase in $\overline{U}$. On the other hand, many corporations have increasingly taken a lenient stance toward letting their employees contribute to open source projects, which will serve to reduce $\overline{U}$.

18. One interpretation of a changing θ would be the learning costs incurred by the community when a licensor chooses an unfamiliar license. Over time, the community learns how the license works and what its likely implications for the development process are, and θ rises.

- Instances where the community derives considerable benefits from being able to tailor the code for its own ends. (Recall this is not a benefit to the licensor, who can always alter the code for his own purposes.)
- Projects that offer higher signaling incentives, whether for ego gratification or for addressing career concerns, provided that these benefits will be more important for the community than for the licensor. (This may be the case, for instance, if the licensor is a corporation.)[19]

Often an open source project may not succeed on a stand-alone basis; rather it may need complementary products in the open source and/or commercial worlds. The choice of license affects the ease with which the different pieces of software can be combined: a point frequently mentioned by advocates of the BSD license, who argue that the GPL and related licenses discourage potential commercial users. This is captured by $\pi(c)$ in our model, which depicts the declining potential commercial benefits from projects with restrictive licenses.

A case in point is the choice of license by programmers trying to get software established as a standard. Although they involve risks of hijacking, unrestrictive licenses make more sense than restrictive ones in such a context. This conjecture leads us to anticipate that projects geared toward the Internet, where setting standards has been particularly important in recent years due to the immaturity of key technologies, might be less likely to have highly restrictive licenses.

Interestingly, the licensing choices may also give rise to "dynamic strategic complementarities" or "dynamic network externalities" among open source licensors. If existing projects in a field have restrictive licenses, the licensor is more likely to choose a restrictive license in anticipation of future user benefits from combining the end results. Conversely, a project with a restrictive license may not flourish in an environment dominated by BSD-licensed projects.[20]

We can also consider the impact of the commercial potential of a project; for instance, projects that run on proprietary operating systems or in commercial

19. If there are large signaling incentives for the licensor as well, then two opposing effects are at work. First, the community is eager to participate, and the licensor can consequentially offer a less restrictive license. But the licensor's benefit from a successful project also increases. Thus he is eager to ensure that the community participates in the project and offers a more restrictive license to ensure their participation.

20. An additional complication is introduced by the asymmetry of the licenses, especially the greater restrictions in the GPL. If a BSD-licensed project wanted to make substantial use of a program (or portion of a program) covered by the GPL, the project leaders would need to obtain permission from the copyright owner (for instance, the Free Software Foundation). Were the leaders of the BSD-licensed program to incorporate the GPL code without permission, their BSD product would effectively be converted into a GPL product. Thus they will be reluctant to add such features. GPL projects, on the other hand, can incorporate elements of (or work alongside) either GPL or BSD programs without subverting their license. Thus, in settings where existing projects have restrictive licenses, founders of new projects may want to also have restrictive licenses in order to ease collaborations. The pressures to choose a particular type of license may be less intense if existing projects have unrestrictive licenses. More generally, the GPL can be seen as serving as an "absorbing state" in a way that the BSD license does not.

firms' user environments. Intuitively such projects should be offered through unrestrictive licenses. But the licensor may make particularly large profits in these cases, and thus may be anxious to ensure that the projects are successes by offering more restrictive licenses.[21]

We will simply derive one limited result in this case, which will prove useful for our empirical analysis. Suppose that

$$\pi(c) = \pi_0 + h(1 - c),$$

where π_0 represents the profits to be made on, say, servicing the open source software, while $h(1 - c)$ corresponds to the potential profits from integrating complementary software with the open source program. (The latter set of profits is thus more sensitive to the nature of the license offered.) Let $c = 1$ correspond to the most restrictive (GPL) license and $c \in [0,1]$ more generally. $V(c = 1)$ is independent of h. By way of contrast, when c is less than one, V is increasing in h, both through the increased profits of the licensor when the community members participate and through the increased level of participation. We therefore conclude that a GPL becomes relatively less attractive than a less restrictive license as h grows. Thus we would anticipate that a GPL or other very restrictive license would be less likely to be selected when the project runs on a proprietary operating system or operates in a commercial environment.

Finally, we address the sample selection bias that is introduced by the licensor's participation constraint $(V \geq \overline{V})$. There is no such bias if the returns from the proprietary project $(\overline{V})$ are independent of the parameters that are relevant to the open source project. We have argued, though, that $\overline{V}$ may be positively correlated with a through the presence of rival software programs. This truncation eliminates some high-a (attractive) projects from the open source domain that are instead kept proprietary. Had they been made into open source projects, these would have received unrestrictive licenses.

We feel that the truncation bias is particularly strong for corporate licensors. Our conversations suggest that in many cases, corporations convert projects into open source ones where they lag behind a competitor and hope to catch up by involving the open source community (or at least to create a competitive threat to their rival in this segment). Thus we could expect the sample selection bias to reinforce the corporation's tendency (derived above) to offer restrictive licenses.

4. The Sample

4.1 Constructing the Sample

The dataset consisted of all software development projects listed on (and for a subset of the analyses, hosted on) SourceForge.net. SourceForge is a free

21. One illustration is the case of Netscape's Mozilla project. The project's initial license was greeted with protests. Worried that the project would not be a success, the project leaders and Netscape replaced it with a more restrictive license (Hammerly, Paquin, and Walton, 1999).

service that since 1999 has offered hosting and project administration tools to software development projects. The site's operations have been funded since its inception by VA Software (formerly known as VA Linux), which at the time of the site's creation was primarily selling computers optimized for Linux. Today, VA has abandoned the hardware business and intends to ultimately earn a profit by selling a version of the SourceForge service to corporations to manage the development of software for internal (proprietary) applications.

SourceForge contained (as of May 2002, when the data was accessed) approximately 39,000 projects. Essentially it accepts listings of (and is willing to host) all projects that conform to the open source definition discussed above, as well as selected projects operating under licenses that are not compliant with that definition.[22] Not all open source projects, however, are hosted on SourceForge. Many of the largest projects instead have their own Web sites. Other projects are hosted at smaller competing sites. These tend to be much smaller, however: Savannah, often referred to as SourceForge's leading competitor, had 790 active projects in May 2002 [http://savannah.gnu.org (accessed September 17, 2002)]. Even when the projects are hosted elsewhere, however, these projects in many cases are often still listed in SourceForge (the reader is simply encouraged to go elsewhere to make a code contribution or report a bug). In cases where a project is listed on SourceForge but hosted elsewhere, we are able to gather the basic data about the project, even if we cannot determine the extent of activity in the project.

The termination of projects is an extremely rare event: projects generally tend to "fade away" rather than being explicitly ceased. SourceForge administrators will remove a project from its database if explicitly requested to by the project administrators, but not otherwise. In total, less than two dozen projects had been removed by the time of the data download. Thus survivorship bias should not be a significant problem.[23]

We accessed the data in two forms:

- The basic data about each project was downloaded from the SourceForge Web site. This information included the stage of development of the project, the environment in which the project operated (e.g., Windows-based systems, handheld devices, Internet applications), the type of license employed, the human language in which the programmers operated, the operating system under which the program ran, and the intended audience. Since project leaders report these data to SourceForge, a natural question relates to their accuracy. An important point to note, though, is that the

22. There are exceptions, however. These include, for instance, projects that involve encryption software that is banned under U.S. law. For a fuller discussion, see http://sourceforge.net/docman/display_doc.php?docid=756&group_id=1 (accessed September 17, 2002).

23. One complication is that a number of projects may have become defunct before SourceForge was established and thus were never included in the database to begin with. To address this concern, we repeat the analyses, eliminating all projects that were added to the database in the first three months of operation. Any sample selection bias from this problem should be greatly reduced in the resulting sample. The results in this subsample are little changed.

project leaders are trying to recruit users to make an extended time commitment to their project. Undertaking a "bait-and-switch" strategy at the time of recruiting new users—for example, by making the project appear to be something other than what it really is—is unlikely to be a positive signal to prospective developers. Only in approximately 40% of the cases, however, was the full information on the project (and especially the license type) available. This reflected the fact that project leaders did not always complete this information at the time the project was established on SourceForge.

- We obtained directly from the SourceForge staff various supplemental measures, including the date at which the project was first posted on SourceForge and the activity at the Web sites (e.g., bug reports submitted and resolved) since the inception of the site in 1999. The latter data were available only for approximately 10,000 projects. In the other instances, the projects could have attracted no activity whatsoever, or else the activity was concentrated on another site.[24]

The two datasets were then merged. The set of projects in the SourceForge database is summarized in the final columns of Tables 1 and 2. In each case, we indicate the distribution for all licenses and for the subset of projects where the site has had active postings from SourceForge users.

Several patterns are evident from these tabulations. First, the dominant role of the GPL is clear. Fully 72% of the licenses are GPLs, and its less constraining cousin, the lesser GPL, represents another 10%. The BSD license, which represents 7% of the sample, is third.[25] Second, the sample is dominated by early stage projects. This dominance is somewhat less pronounced in the tabulation of projects with contributions: not surprisingly, the youngest projects have garnered the fewest contributions to date. Third, the sample is dominated by projects in English, oriented toward end-users and developers, and geared to two families of operating systems (the POSIX family, which includes Linux, BSD, and Sun's Solaris, and Microsoft) or else are independent of any operating system.

24. Other concerns that might be raised about the performance measures are not borne out. Because of the extent of the coordination costs, even projects with multiple sites tend to have all (or virtually all) the contributions focused on a single one of those sites. Switching projects from SourceForge to another site appears very rare, in large part due to the "lock-in effects" that SourceForge enjoys. See, for instance, the discussion in http://www.advogato.org/article/376.html (accessed September 17, 2002).

25. These tabulations are not weighted (i.e., each project is counted equally). We do not have a count of the number of lines of code in the project, which might be a natural weighting. We do have, however, the total number of problems ("bugs") reported to the SourceForge depository. While this measure is not as satisfactory (some projects operate code depositories that are not part of the SourceForge site, and thus appear to have little activity, but are actually quite vital), it may nonetheless be a reasonable proxy. The results of the weighted analysis suggest that the GPL is not as dominant: 63% of the weighted projects have GPL s, 11% have lesser GPLs, and 11% have BSD licenses.

Table 2. Characteristics of the SourceForge Sample

Intended Audience		Topic		Natural Language		Environment		Operating System		Development Stage	
Panel A: Percentage distribution of the entire sample											
End Users/Desktop	55.8	Communications	16.2	English	95.8	Console (Text)	30.7	POSIX	57.1	Planning	31.8
Developers	64.1	Security	3.2	French	5.7	X11	29.0	Microsoft	28.6	Pre-Alpha	21.3
System Administrators	24.5	Software Dvlpmt.	17.6	Spanish	3.1	MS Windows	24.1	OS/2	0.2	Alpha	18.9
Other	14.2	Desktop Environ.	4.5	Japanese	1.1	Other	13.5	MacOS	3.5	Beta	22.0
		Text Editors	2.8	German	9.2	Internet	31.4	BeOS	0.9	Production/Stable	16.8
		Database	6.9	Russian	1.3	No Input/Output	10.3	OS Independent	36.8	Mature	1.8
		Education	3.2			Cocoa (MacOS)	1.0	Other	2.0		
		Internet	24.0			Handhelds/PDAs	0.3	PDA Systems	0.1		
		Scientific/Enging.	8.3								
		Multimedia	11.5								
		Office/Business	5.1								
		System Tasks	19.8								
		Printing	0.5								
		Terminals	0.7								
		Other	3.1								
		Games, et al.	15.4								
Panel B: Percentage distribution of sample with activity data											
End Users/Desktop	53.1	Communications	15.3	English	96.7	Console (Text)	32.7	POSIX	57.9	Planning	8.1
Developers	66.5	Security	3.6	French	5.4	X11	30.2	Microsoft	29.8	Pre-Alpha	12.4
System Administrators	28.5	Software Dvlpmt.	21.2	Spanish	2.9	MS Windows	25.1	OS/2	0.3	Alpha	22.0
Other	12.6	Desktop Environ.	5.1	Japanese	1.2	Other	13.1	MacOS	3.7	Beta	36.2
		Text Editors	3.4	German	8.7	Internet	27.9	BeOS	1.0	Production/Stable	32.0
		Database	7.1	Russian	1.7	No Input/Output	10.6	OS Independent	36.4	Mature	3.3
		Education	2.7			Cocoa (MacOS)	1.2	Other	1.9		

Continued

Table 2. *Continued*

Intended Audience		Topic		Natural Language		Environment		Operating System		Development Stage	
		Internet	24.1			Handhelds/PDAs	0.3	PDA Systems	0.2		
		Scientific/Enging.	9.5								
		Multimedia	14.3								
		Office/Business	4.6								
		System Tasks	20.4								
		Printing	0.9								
		Terminals	0.9								
		Other	2.4								
		Games, et al.	12.2								

The table summarizes the percentage of projects classified along a number of dimensions of the 38,610 projects included in the SourceForge database in May 2002. Panel A presents the distribution for the entire sample. Panel B presents the distribution for the subset of 9257 observations where SourceForge has data on software contributions. Some projects may be classified into multiple categories.

Table 3. Cross-Tabulation of Intended Audience and Project Topic

	Intended audience			
	End users/desktop	Developers	System administrators	Other
Communications	13.3%	9.2%	13.5%	11.6%
Security	1.8%	1.9%	5.0%	2.3%
Software Dvlpmt.	4.9%	18.4%	6.8%	7.5%
Desktop Environ.	4.7%	2.5%	1.9%	2.0%
Text Editors	2.3%	2.3%	1.2%	1.5%
Database	4.2%	5.4%	6.2%	4.5%
Education	2.9%	1.8%	1.3%	4.8%
Internet	14.5%	17.3%	25.2%	18.3%
Scientific/Enging.	5.9%	6.3%	1.3%	10.2%
Multimedia	10.2%	7.5%	2.2%	6.6%
Office/Business	4.9%	2.9%	3.0%	4.8%
System Tasks	11.6%	13.0%	27.3%	11.3%
Printing	0.5%	0.3%	0.4%	0.3%
Terminals	0.6%	0.5%	0.8%	0.5%
Other	2.8%	1.9%	1.3%	3.5%
Games, et al.	14.7%	8.7%	2.6%	10.2%

The table summarizes the distribution of projects classified along two dimensions of the 38,610 projects included in the SourceForge database in May 2002. Each column indicates the percentage of projects geared to each intended audience with that particular topic.

A natural concern is the extent to which the measures are colinear: in other words, the extent to which the characteristics of the projects are highly correlated with each other. Table 3 provides a cross-tabulation of project topic and intended audience. To be sure, there is some clustering: for instance, projects geared toward system administrators disproportionately involve security and systems tools (from which they presumably derive greater private benefits from tailoring the projects to their needs). But certainly, a considerable degree of diversity exists in this and other comparisons.

Table 4 provides another overview of the sample. It summarizes the 25 most active projects, based on SourceForge's ranking of the activity percentile as of May 2002. It also reports the nature of the project and the license, employing the categorization of highly restrictive and restrictive licenses discussed in Section 2. (In some cases, software is made available under multiple licenses of different types.) The considerable diversity of license and project types is apparent.

4.2 Mapping Between Theory and Data

As discussed above, we hypothesize that in settings where software has limited community appeal, the leader will need to offer a restrictive license in order to induce participation. We anticipate that this will be the case in three instances:

- If the community distrusts the licensor.
- If the benefits from tailoring the code for particular applications are weak.

Table 4. The 25 Most Active Projects

		License type		
Project name	Activity percentile	Highly restrictive	Restrictive	Description
JBoss.org	100	No	Yes	Application server implemented in Java.
phpMyAdmin	99.99	Yes	Yes	Tool handling administration of web-based MySQL databases.
Gaim	99.98	Yes	Yes	Instant messenger application supporting multiple protocols.
DC++	99.97	Yes	Yes	C++ version of the Direct Connect client.
PCGen	99.96	No	Yes	Role-playing game character generator and maintenance program.
Compiere ERP + CRM	99.95	No	Yes	Customer management, supply chain, and accounting tools for small business.
Dev-C++	99.94	Yes	Yes	Full-feature Integrated Developer Environment for Windows.
Arianne RFG	99.93	Yes	Yes	Engine that allows easy creation of multi-player online games.
Firewall Builder	99.92	Yes	Yes	Object-oriented interface for various firewall programs.
ScummVM	99.91	Yes	Yes	Cross-platform translator for point-and-click adventure engines.
MiKTeX	99.90	Partial	Partial	Implementation of TeX & Friends in Windows.
SquirrelMail	99.89	Yes	Yes	PHP-based Web e-mail client.
Bochs x86 PC Emulator	99.88	No	Yes	Portable x86 PC emulation software package for workstations.
Xdoclet	99.87	No	No	Tool to generate Javadocs @tags and source code.
Marathon	99.86	No	Yes	Graphical development tool for Firebird and InterBase databases.
CDex	99.85	Yes	Yes	"Ripper" that allows extraction of digital audio data from audio CDs.
HSQL Database Engine	99.84	No	No	Relational database engine written in Java.
BlackNova Traders	99.83	No	No	Web-based, multi-player space exploration game.
FileZilla	99.82	Yes	Yes	Feature-rich FTP client for Windows.
Miranda IM Client	99.81	Yes	Yes	Instant messaging client for Windows.
Fink	99.80	Yes	Yes	Adapting Unix open source software to Darwin and Mac OSx.
Fluxbox	99.79	No	No	X.11 window management program.
TUTOS	99.78	Yes	Yes	Web-based groupware program to manage teams, events, projects, etc.
zenTrack	99.77	Yes	Yes	Complete project management, bug tracking, and tech support log system.
Pure-FTPd	99.76	Yes	Yes	FTP server, based on Troll-FTPd.

The table indicates, for the 25 most active projects in the SourceForge dataset, the name of the project, its activity percentile, the license type, and a description of the project.

- If ego gratification and career concerns incentives do not have much power, as the audience mostly does not look at the code and is not composed of the programmers' peers.

In addition, under certain additional assumptions, we show that a highly restrictive license is unlikely to be chosen when the project runs a proprietary operating system or operates in a commercial environment.

In this section we review the independent variables we employ in the analysis, and seek to understand the predicted effects for each.

4.2.1 Intended Audience. In our earlier work (2002), we argue that an important driver of the decision to participate in open source projects is the career enhancement and ego gratification that results in demonstrating one's programming prowess to sophisticated peers. Thus we hypothesize that code aimed at developers, and to a lesser extent, system administrators, is more likely to belong to the "strong community appeal" category. This reasoning suggests that code aimed at developers is more likely to be licensed under a permissive license than code oriented toward unsophisticated end users.

4.2.2 Topic. On a similar note, we believe that projects on topics likely to be geared toward sophisticated peers are more likely to have permissive licensing terms, while those geared toward end-users will be less permissive. Among the categories identified by SourceForge, "Software Development" seems unambiguously in the former class. "Education," "Games," and "Office/Business" fall clearly in the latter. The other categories cannot be as clearly parsed, or else seem largely geared to the intermediate category of system administrators.[26]

4.2.3 Natural Language. We anticipate that projects whose natural language is not English will have a harder time generating community appeal because the pool of potential contributors is considerably smaller. As a result, we expect that projects with another native language will be more likely to have a restrictive license.

It should be noted that Japan is a special case, because the only localized version of the SourceForge database (true both at the time of the data extraction and in 2004) is http://sourceforge.jp. As a result, there appears to be a significant selection effect at work, which leads to only a modest fraction of Japanese projects (presumably the ones with the greatest appeal to the open source community) included in the main SourceForge database. Thus we cannot really compare the coefficients on the Japanese variable with the others due to this selection effect.

26. For instance, "printing" includes both device drivers geared toward end-users as well as libraries for developers.

4.2.4 Environment and Operating System. We employ two proxies that should imperfectly capture the programs' technical features: the environment for which the program is intended and the operating system on which it runs.[27] The theoretical analysis—as well as the broader considerations discussed in Section 3—suggests that projects operating in more commercial settings will employ less restrictive licenses.

4.2.5 Development Stage. The final set of variables is intended as a control. These projects are of different "vintages": while some have been in existence for a considerable period of time and are well developed, others are recent projects and in an early stage. As highlighted above, the choice of license type may be influenced by many considerations, including shifts in the opportunity cost of participating in such projects ($\overline{U}$), network effects, and shifting ideologies. We might expect that the strength of these considerations might change over time. As a result, it might be anticipated that the measures of development stage might capture the shifting propensity of different vintages of project leaders to choose particular licenses.[28]

5. The Determinants of Open Source Licenses

We then examine the determinants of the license types employed in these contracts. We first explore the individual licensing components and then use an index of license scope.

We first summarize the distribution of projects along two measures of license scope that we discussed in Section 2: whether the license is restrictive or not and whether it is highly restrictive or not. These tabulations are challenging because of the complexity of some situations. Some projects operate under multiple licenses: in these instances, different sections of the code may be under different licenses, or the contributor may be able to choose the license he wishes to govern his contribution. In other cases, a single license may allow a user to choose the degree of protection he wishes to have. We thus code each project as to whether all or some of the code contributed was subject to restrictive or highly restrictive provisions.

Table 5 highlights several patterns:

- Consistent with the framework in Section 3, highly restrictive licenses are more common for applications geared toward end-users, but are

27. The environment refers to the graphical user interface for the user's desktop. In addition to capabilities provided by a window manager (i.e., the ability to move, resize, and hide windows), a desktop environment usually also includes such elements as a file manager, a program manager, and other utilities. An operating system is the software responsible for allocating system resources, including memory, processor time, disk space, and peripheral devices such as printers, modems, and the monitor. All application programs use the operating system to gain access to these system resources as they are needed.

28. We will also address this vintage question in another way below, by examining projects that were added to SourceForge at different times in separate regressions.

significantly less common for those applications aimed toward software developers. Highly restrictive licenses are also more common for projects geared to systems administrators, which may reflect the weaker community appeal of these efforts.

- Also consistent with the above framework, applications that are consumer oriented (e.g., desktop tools and games) are substantially more likely to have highly restrictive licenses. Those geared to the software development process are much less so. Similarly products geared to technical users (e.g., scientific and engineering programs and database software) are less likely to have highly restrictive licenses.
- Highly restrictive licenses are generally much more common for projects whose natural language is other than English. (Japanese, as noted above, is a special case.) These results are also consistent with our theoretical predictions.
- Highly restrictive licenses are less common for projects operating in commercial environments such as Microsoft Windows or Apple's Cocoa. But projects operating in the X11 environment—a network-transparent window system developed at MIT which runs on a wide range of computing and graphics machines—are more likely to be highly restrictive. While this and the following result are consistent with our theory, our interpretation must be cautious.
- Highly restrictive licenses are significantly more common for projects that run under the open source-based POSIX family of operating systems, as opposed to proprietary ones (or those which are operating system independent).
- Highly restrictive licenses are less common for more mature projects. As we will discuss below, this pattern appears to reflect a "vintage effect": it may have been more common for older projects to employ licenses other than the GPL. In part, ideological considerations may have played a role in this trend. During the late 1990s, the stridency of arguments for the GPL by its proponents seemed to intensify. Alternatively, the opportunity cost for programmers may have shifted over time.

When we examine in Table 6 the presence of restrictive provisions, we find a similar pattern. Exceptions include the absence of any significant pattern involving products geared to system administrators, and a somewhat different mixture of topics where restrictive provisions are commonplace.

Tables 7 and 8 then examine these patterns in a regression framework. Reflecting the fact that the dependent variable is in each case a dummy, we employ a probit specification. For each class of variables we delete one of the independent variables from the specification: the dummy variables denoting projects in the planning stage, those operating in a console (text) environment, those geared toward other audiences, those whose natural language is English, those geared toward another operating system, and those with another topic.

Table 5. Tabulation of Characteristics of Projects With and Without Highly Restrictive License Provisions

	All Licenses Highly Restrictive?		Some Licenses Highly Restrictive?	
	Yes	No	Yes	No
Intended audience				
End-users/desktop	62.3	***42.0	62.0	***40.2
Developers	57.3	***78.5	58.3	***78.4
System administrators	29.5	***23.3	29.6	***22.2
Other	14.7	***12.8	14.8	***12.2
Topic				
Communications	17.0	***14.1	16.9	***14.1
Security	3.2	3.1	3.3	2.9
Software development	12.2	***29.6	12.8	***30.3
Desktop environment	4.9	***3.9	4.9	***3.7
Text editors	2.8	3.0	2.8	3.0
Database	6.6	***7.7	6.6	***7.7
Education	3.3	*2.9	3.3	3.0
Internet	24.1	23.9	23.9	24.4
Scientific/engineering	7.9	***9.3	8.0	***9.4
Multimedia	11.4	12.0	11.5	11.7
Office/business	5.5	***4.4	5.5	***4.3
System tasks	20.4	***18.6	20.8	***17.4
Printing	0.4	**0.7	0.5	0.6
Terminals	0.8	0.6	0.8	**0.5
Other	3.1	3.2	3.0	3.3
Games, etc.	16.6	***12.3	16.5	***12.1
Natural language				
English	95.4	***97.1	95.5	***97.2
French	6.0	***4.8	6.0	***4.7
Spanish	3.5	***2.4	3.4	***2.3
Japanese	0.8	***1.7	0.9	***1.6
German	10.4	***6.6	10.2	***6.5
Russian	1.2	*1.5	1.3	1.5
Environment				
Console (text)	30.3	***32.8	31.0	31.3
X11	31.2	***24.7	31.6	***22.9
MS Windows	22.7	***26.5	22.9	***26.5
Other	10.9	***19.5	11.4	***19.4
Internet	31.2	31.0	30.9	31.9
No input/output	9.5	12.5	9.8	12.1
Cocoa (MacOS)	0.9	***1.3	0.9	***1.4
Handhelds/PDAs	0.3	0.4	0.3	0.4
Operating system				
POSIX	61.3	***48.9	61.6	***46.6
Microsoft	27.0	***31.6	27.2	***31.6
OS/2	0.2	0.2	0.2	0.2
MacOS	2.8	***5.1	2.9	***5.2

Continued

Table 5. *Continued*

	All Licenses Highly Restrictive?		Some Licenses Highly Restrictive?	
	Yes	No	Yes	No
BeOS	0.7	***1.4	0.8	***1.4
OS independent	33.1	***44.8	33.1	***46.1
Other	1.8	***2.5	1.9	2.2
PDA Systems	0.2	0.1	0.2	0.1
Development Stage				
Planning	32.2	***28.9	32.4	***28.1
Pre-Alpha	21.7	**20.3	21.8	***20.0
Alpha	18.6	***20.1	18.8	**19.8
Beta	21.8	***23.5	22.0	**23.3
Production/stable	16.3	***18.8	16.5	***18.6
Mature	1.4	***2.7	1.5	***2.6

The sample consists of 38,610 projects included in the SourceForge database in May 2002. The table summarizes the percentage of projects with and without highly restrictive licensing provisions, classified along a number of dimensions. Because some projects are licensed under multiple licenses, the projects are separated as to whether all licenses are highly restrictive or not, and whether some licenses are highly restrictive or not. Some projects may be classified into multiple categories. The significance level from a chi-squared test is reported in each case where the null hypothesis of no difference is rejected.
*, **, ***Significant at the 10%, 5%, and 1% confidence levels, respectively.

The primary differences in the results in Table 7 from those in the univariate analyses are as follows:

- Software geared toward developers is sharply different from that geared toward other users, being much less likely to have highly restrictive licenses.
- Among the projects less likely to have highly restrictive licenses are those related to software development, desktop applications, the Internet, multimedia, and printing, consistent with the arguments above.
- Projects whose natural language is Japanese are far less likely to have highly restrictive licenses, while German and Spanish ones are much more likely to have them.

The results in Table 8 are similar, with the exception again of no significant pattern involving products geared to system administrators and a somewhat different mixture of topics where restrictive licenses are commonplace.

These effects are not only statistically significant, but economically meaningful as well. Consider, for instance, the first regression in Table 7. A project in the planning stages (the omitted case) has a 12% higher predicted probability of all licenses being highly restrictive than one in the mature stages. A project geared toward individual end-users has a 23% higher probability of all licenses being highly restrictive than one oriented toward developers.

The regression analysis in Table 9 looks at restrictive and highly restrictive licenses in a single specification. To do this, we employ indexes, which

Table 6. Tabulation of Characteristics of Projects With and Without Restrictive License Provisions

	All licenses restrictive?		Some licenses restrictive?	
	Yes	No	Yes	No
Intended audience				
End-users/desktop	58.2	***46.3	58.1	***45.6
Developers	61.8	***73.1	62.2	***72.5
System administrators	27.5	27.7	27.7	26.6
Other	14.0	14.6	14.1	14.0
Topic				
Communications	16.3	*15.2	16.3	15.2
Security	3.1	**3.7	3.1	3.4
Software development	15.8	***25.5	16.0	***25.5
Desktop environment	4.8	***3.4	4.8	***3.3
Text editors	2.8	3.2	2.8	3.1
Database	6.8	7.2	6.8	7.3
Education	3.1	3.5	3.1	*3.6
Internet	23.5	***26.1	23.6	***26.2
Scientific/engineering	8.6	**7.5	8.5	*7.7
Multimedia	12.0	***10.0	12.0	***9.8
Office/business	5.3	**4.6	5.3	**4.5
System tasks	19.8	20.1	20.0	19.0
Printing	0.5	*0.7	0.5	0.6
Terminals	0.8	0.6	0.8	0.6
Other	2.9	***3.8	2.9	***3.9
Games, etc.	15.9	***12.7	15.8	***12.7
Natural language				
English	95.6	***97.1	95.7	***97.1
French	5.9	***4.7	5.9	***4.4
Spanish	3.3	***2.3	3.3	***2.2
Japanese	0.9	***1.9	0.9	***1.7
German	10.1	***5.7	10.0	***5.6
Russian	1.2	*1.6	1.3	1.6
Environment				
Console (text)	30.2	***34.6	30.6	***33.3
X11	31.2	***21.3	31.1	***20.6
MS Windows	23.7	24.6	23.7	24.3
Other	12.3	***18.3	12.6	***17.7
Internet	30.7	***33.1	30.7	***33.3
No input/output	9.8	***12.8	10.0	12.3
Cocoa (MacOS)	0.9	***1.4	0.9	***1.4
Handhelds/PDAs	0.3	0.3	0.3	0.3
Operating system				
POSIX	59.5	***49.0	59.6	***47.2
Microsoft	27.9	***30.7	28.0	***30.4
OS/2	0.2	0.3	0.2	0.2
MacOS	3.0	***5.7	3.1	***5.7

Continued

Table 6. *Continued*

	All licenses restrictive?		Some licenses restrictive?	
	Yes	No	Yes	No
BeOS	0.8	***1.4	0.8	***1.5
OS independent	35.1	***43.4	35.2	***44.0
Other	1.8	***2.9	1.9	***2.5
PDA systems	0.2	0.2	0.2	0.1
Development stage				
Planning	31.6	***29.6	31.7	***29.0
Pre-Alpha	21.5	*20.3	21.6	***19.8
Alpha	18.9	19.8	19.0	19.7
Beta	22.2	22.7	22.3	22.2
Production/stable	16.5	***19.5	16.6	***19.0
Mature	1.5	***3.1	1.6	***2.8

The sample consists of 38,610 projects included in the SourceForge database in May 2002. The table summarizes the percentage of projects with and without restrictive licensing provisions, classified along a number of dimensions. Because some projects are licensed under multiple licenses, the projects are separated as to whether all licenses are restrictive or not, and whether some licenses are restrictive or not. Some projects may be classified into multiple categories. The significance level from a chi-squared test is reported in each case where the null hypothesis of no difference is rejected.

*, **, *** Significant at the 10%, 5%, and 1% confidence levels, respectively.

measure whether the project has various licensing provisions. Because of the ambiguities surrounding the interpretation of cases where there are alternative licenses, we proceed in two ways. In the first regression, the index takes on the value 4 if all licenses are highly restrictive; 3 if some are highly restrictive; 2 if all licenses are restrictive but none are highly restrictive; 1 if some are restrictive but none are highly restrictive; and 0 otherwise. In the second regression, the index takes on the value 2 if all licenses are highly restrictive; 1 if all are restrictive and some (but not all) are highly restrictive; and 0 otherwise.

We estimate ordered logit regressions because of the nature of the dependent variable. In an ordered logit specification, a license that was rated as a “4” would be treated as being more restrictive than one rated as a “2,” but not necessarily twice as much so. The findings in Table 9 are largely consistent with the analyses reported above, particularly those in Table 8.

One concern with the analysis in Table 9 is the presence of projects with multiple licenses. We explore the robustness of the results in unreported regressions. Rather than denoting projects that have “all highly restrictive” and “some highly restrictive” licenses, we treat the cases with multiple licenses in two different ways. We first reestimate the equations, eliminating all projects that have multiple licenses. We also rerun the regressions employing the maximum degree of restrictiveness of any license. The results are little changed in either case.

We also undertook an analysis that attempted to control for the age of the open source project. As noted above, we were concerned that a survival effect might be at work: the characteristics of older projects might be different from

Table 7. Regression Analysis of Characteristics of Projects With and Without Highly Restrictive License Provisions

	Dependent variable			
	All Licenses Highly Restrictive?		Some Licenses Highly Restrictive?	
	Coefficient	Standard error	Coefficient	Standard error
Intended audience				
End-Users/desktop	0.32	***0.03	0.37	***0.03
Developers	–0.24	***0.03	–0.18	***0.03
System administrators	0.14	***0.03	0.16	***0.03
Topic				
Communications	–0.01	0.03	–0.003	0.03
Security	–0.08	0.06	–0.06	0.06
Software development	–0.40	***0.03	–0.36	***0.03
Desktop environment	–0.15	***0.06	–0.12	**0.06
Text editors	–0.01	0.07	–0.01	0.07
Database	0.005	0.04	0.03	0.05
Education	–0.09	0.06	–0.10	*0.06
Internet	–0.08	***0.03	–0.08	**0.03
Scientific/engineering	–0.08	*0.04	–0.06	0.04
Multimedia	–0.16	***0.04	–0.12	***0.04
Office/business	–0.04	0.05	–0.001	0.05
System tasks	–0.04	0.03	0.01	0.03
Printing	–0.44	***0.17	–0.43	**0.17
Terminals	–0.03	0.13	0.09	0.14
Games, etc.	0.05	0.04	0.07	*0.04
Natural language				
French	0.08	*0.05	0.11	***0.05
Spanish	0.18	***0.07	0.18	***0.07
Japanese	–0.46	***0.10	–0.38	***0.11
German	0.23	***0.04	0.19	***0.04
Russian	0.08	0.10	0.10	0.10
Environment				
X11	0.05	0.03	0.10	***0.04
MS Windows	–0.07	*0.04	–0.07	*0.04
Other	–0.32	***0.03	–0.30	***0.03
Internet	–0.03	0.04	–0.02	0.03
No input/output	–0.28	***0.04	–0.23	***0.04
Cocoa (MacOS)	–0.06	0.10	–0.15	0.10
Handhelds/PDAs	–0.21	0.19	–0.20	0.19
Operating system				
POSIX	0.16	***0.03	0.21	***0.03
Microsoft	–0.15	***0.03	–0.15	***0.04
OS/2	–0.01	0.25	–0.04	0.26
MacOS	–0.36	***0.06	–0.32	***0.06
BeOS	–0.27	**0.12	–0.24	**0.12
OS independent	–0.16	***0.03	–0.14	***0.03

Continued

Table 7. *Continued*

	Dependent variable			
	All Licenses Highly Restrictive?		Some Licenses Highly Restrictive?	
	Coefficient	Standard error	Coefficient	Standard error
PDA systems	0.23	0.26	0.33	0.26
Development stage				
Pre-Alpha	−0.06	*0.03	−0.01	0.03
Alpha	−0.07	**0.03	−0.02	0.03
Beta	−0.09	***0.03	−0.04	0.03
Production/stable	−0.15	***0.03	−0.11	***0.03
Mature	−0.34	***0.08	−0.28	***0.08
Constant	0.80	***0.05	0.72	***0.05
χ^2 statistic	1,580.35		1,494.92	
p-value	0.000		0.000	
Log-likelihood	−8,580.97		−8,141.81	
Number of observations	15,509		15,509	

The sample consists of 38,610 projects included in the SourceForge database in May 2002. The dependent variable is a dummy denoting whether the project has highly restrictive licensing provisions. Because some projects are licensed under multiple licenses, the projects are separated as to whether all licenses are highly restrictive or not, and whether some licenses are highly restrictive or not. The independent variables include dummy variables capturing various features of the open source projects. All regressions employ probit specifications.
*, **, *** Significant at the 10%, 5%, and 1% confidence levels respectively.
In each regression, the following variables are omitted: the dummy variables denoting projects in the planning stage, those operating in a console (text) environment, those geared toward other audiences, those whose natural language is English, those geared toward another operating system, and those with another topic.

others. This effect might lead to the conclusion that a given feature affected the choice of license, when it was actually the age that was critical.

While we do not know the date at which the project was initiated, we do have a proxy for this measure: when the project was added to the SourceForge database. (Because the database only began operations in 1999, this measure does not allow us to identify the oldest projects.) We employ this measure in several ways. Table 10 shows the most direct approach. We reestimate the regression reported in the first column of Table 7, first restricting the sample to the oldest projects (those added to the SourceForge database in its first year of operations) and the youngest (those added in 2002).

The patterns relating to stage of development disappear in these regressions, underscoring the suggestion that this measure may be capturing a vintage effect. But at the same time, the key explanatory variables differ little across the time periods. Projects geared toward end-users tend to have highly restrictive licenses, while those oriented toward developers are less likely to have them. The types of projects that are likely to be attractive to consumers—such as games—are more likely to have highly restrictive licenses. Finally, projects that are designed to run on commercial operating systems are less likely to have highly restrictive licenses.

Table 8. Regression Analysis of Characteristics of Projects With and Without Restrictive License Provisions

	Dependent variable			
	All licenses restrictive?		Some licenses restrictive?	
	Coefficient	Standard error	Coefficient	Standard error
Intended audience				
End-users/desktop	0.15	***0.03	0.18	***0.03
Developers	–0.07	***0.03	–0.05	0.03
System administrators	0.02	0.03	0.05	0.03
Topic				
Communications	0.03	0.04	0.03	0.04
Security	–0.11	*0.06	–0.04	0.07
Software development	–0.20	***0.04	–0.15	***0.04
Desktop environment	–0.09	0.06	–0.07	0.07
Text editors	–0.08	0.07	–0.06	0.07
Database	0.08	0.05	0.07	0.05
Education	–0.15	**0.07	–0.13	*0.07
Internet	–0.07	**0.03	–0.06	*0.03
Scientific/engineering	0.10	**0.05	0.09	*0.05
Multimedia	0.01	0.04	0.05	0.04
Office/business	0.01	0.06	0.06	0.06
System tasks	–0.03	0.03	0.01	0.04
Printing	–0.33	*0.17	–0.31	*0.18
Terminals	0.12	0.14	0.19	0.15
Games, etc.	0.06	0.04	0.09	**0.04
Natural Language				
French	0.07	0.05	0.13	**0.06
Spanish	0.17	**0.07	0.18	**0.08
Japanese	–0.44	***0.11	–0.30	***0.11
German	0.27	***0.05	0.26	***0.05
Russian	0.01	0.10	0.05	0.11
Environment				
X11	0.18	***0.04	0.19	***0.04
MS Windows	–0.05	0.04	–0.04	0.04
Other	–0.22	***0.04	–0.18	***0.04
Internet	–0.04	0.03	–0.02	0.03
No input/output	–0.18	***0.04	–0.14	***0.04
Cocoa (MacOS)	–0.07	0.11	–0.08	0.11
Handhelds/PDAs	–0.01	0.21	–0.02	0.21
Operating system				
POSIX	0.14	***0.04	0.17	***0.04
Microsoft	–0.07	**0.04	–0.05	0.04
OS/2	–0.31	0.25	–0.41	0.26
MacOS	–0.34	***0.07	–0.34	***0.07
BeOS	–0.24	*0.12	–0.28	**0.13
OS independent	–0.06	*0.04	–0.04	0.04
PDA systems	0.09	0.27	0.27	0.30

Continued

Table 8. *Continued*

	Dependent variable			
	All licenses restrictive?		Some licenses restrictive?	
	Coefficient	Standard error	Coefficient	Standard error
Development Stage				
Pre-Alpha	−0.04	0.03	0.001	0.03
Alpha	−0.07	**0.03	−0.03	0.03
Beta	−0.04	0.03	0.003	0.03
Production/stable	−0.12	***0.03	−0.09	**0.04
Mature	−0.33	***0.08	−0.25	***0.09
Constant	0.96	***0.05	0.86	***0.05
χ^2 statistic	587.86		527.80	
p-value	0.000		0.000	
Log-likelihood	−7,168.42		−6,733.77	
Number of observations	15,509		15,509	

The sample consists of 38,610 projects included in the SourceForge database in May 2002. The dependent variable is a dummy denoting whether the project has restrictive licensing provisions. Because some projects are licensed under multiple licenses, the projects are separated as to whether all licenses are restrictive or not, and whether some licenses are restrictive or not. The independent variables include dummy variables capturing various features of the open source projects. All regressions employ probit specifications.

*, **, *** Significant at the 10%, 5%, and 1% confidence levels, respectively.

In each regression, the following variables are omitted: the dummy variables denoting projects in the planning stage, those operating in a console (text) environment, those geared toward other audiences, those whose natural language is English, those geared toward another operating system, and those with another topic.

In unreported regressions, we explore the impact of time in a variety of ways. We employ dummy variables denoting the year the project was added to the SourceForge database as independent variables. We also include interaction terms between the date of inclusion and the other key independent variables. These changes have only a very modest effect on the results.

One prediction offered in Section 3 was that projects that were borne out of corporations should differ from other ones. We suggested that in cases where a corporation made its own code available to third parties, the license type should be particularly constraining. We examine this possibility in an exploratory analysis. From a careful examination of news stories and corporate Web sites, we identified 51 entries where we could unambiguously determine that the project originated with proprietary software developed by a corporation. While the number of such cases is modest, such an approach allows us to at least tentatively explore this theoretical suggestion.

As Table 11 reports, projects that involve software developed in a corporate setting are likely to have more restrictive licenses. While the effects are in the predicted direction, and the magnitude of the coefficients are in some cases substantial, the results never become statistically significant. Nonetheless, the results are at least suggestive.

We also address the concern that the inactive projects (ones where no code contributions are made to the SourceForge site) listed on the site are identified in

Table 9. Regression Analysis of Characteristics of Projects With and Without Various License Provisions

	Dependent variable			
	Five-part index		Three-part index	
	Coefficient	Standard error	Coefficient	Standard error
Intended audience				
End-users/desktop	0.49	***0.04	0.46	***0.04
Developers	−0.37	***0.04	−0.38	***0.05
System administrators	0.21	***0.05	0.19	***0.05
Topic				
Communications	−0.003	0.05	0.0004	0.05
Security	−0.13	0.10	−0.17	0.10
Software development	−0.51	***0.05	−0.52	***0.05
Desktop environment	−0.21	**0.09	−0.23	**0.09
Text editors	−0.05	0.11	−0.05	0.11
Database	0.04	0.07	0.04	0.07
Education	−0.18	*0.11	−0.18	*0.11
Internet	−0.13	***0.05	−0.13	***0.05
Scientific/engineering	−0.07	0.07	−0.06	0.07
Multimedia	−0.19	***0.06	−0.19	***0.06
Office/business	−0.03	0.09	−0.05	0.09
System tasks	−0.04	0.05	−0.06	0.05
Printing	−0.70	***0.26	−0.70	***0.26
Terminals	0.03	0.21	0.004	0.21
Games, etc.	0.12	*0.06	0.10	*0.06
Natural language				
French	0.16	*0.08	0.14	*0.08
Spanish	0.33	***0.11	0.33	***0.11
Japanese	−0.64	***0.16	−0.74	***0.16
German	0.38	***0.07	0.39	***0.07
Russian	0.13	0.15	0.08	0.15
Environment				
X11	0.14	**0.06	0.14	**0.06
MS Windows	−0.10	0.06	−0.11	*0.06
Other	−0.46	***0.05	−0.47	***0.05
Internet	−0.05	0.05	−0.05	***0.05
No input/output	−0.40	***0.06	−0.42	***0.06
Cocoa (MacOS)	−0.14	0.16	−0.11	0.16
Handhelds/PDAs	−0.26	0.29	−0.23	0.29
Operating system				
POSIX	0.29	***0.05	0.26	***0.05
Microsoft	−0.22	***0.06	−0.22	***0.06
OS/2	−0.24	0.42	−0.20	0.42
MacOS	−0.58	***0.10	−0.59	***0.10
BeOS	−0.43	**0.18	−0.41	**0.19
OS independent	−0.21	***0.05	−0.22	***0.05
PDA systems	0.39	0.42	0.31	0.42

Continued

Table 9. *Continued*

	Dependent variable			
	Five-part index		Three-part index	
	Coefficient	Standard error	Coefficient	Standard error
Development Stage				
Pre-Alpha	−0.06	0.05	−0.08	*0.05
Alpha	−0.09	*0.05	−0.12	**0.05
Beta	−0.10	**0.05	−0.12	**0.05
Production/stable	−0.21	***0.05	−0.24	***0.05
Mature	−0.51	***0.13	−0.56	***0.13
χ^2 statistic	1,393.70		1,352.98	
p-value	0.000		0.000	
Log-likelihood	−13,064.97		−11,662.26	
Number of observations	15,509		15,509	

The sample consists of 38,610 projects included in the SourceForge database in May 2002. The dependent variable indexes whether the project has various licensing provisions. In the first regression, the index takes on the value 4 if all licenses are highly restrictive; 3 if some are highly restrictive; 2 if all licenses are restrictive; 1 if some are restrictive; and 0 otherwise. In the second regression, the index takes on the value 2 if all licenses are highly restrictive; 1 if all are restrictive; and 0 otherwise. The independent variables include dummy variables capturing various features of the open source projects. All regressions employ ordered logit specifications.

*, **, *** Significant at the 10%, 5% and 1% confidence levels, respectively.

In each regression, the following variables are omitted: the dummy variables denoting projects in the planning stage, those operating in a console (text) environment, those geared toward other audiences, those whose natural language is English, those geared toward another operating system, and those with another topic.

a manner that introduces some biases. We rerun the regressions reported here, restricting the sample to the approximately 10,000 observations with code contributions. We also repeat the analysis, weighting the observations by a number of activity measures: the number of bugs reported, the number of active developers, and the percentile of activity of the project. While, as discussed in the footnote 25, the mixture of licenses employed changes somewhat when such weights are employed, the magnitude and significance of the key independent variables are little changed.

Another concern was that the ideological considerations noted in Section 3 may distort the decisions being made. To partially address this concern, we reran the regressions reported in Table 9, eliminating those with BSD licenses and GPLs, the two licenses whose use has attracted the most polarized debate. The results remained similar: for instance, those projects geared toward end-users and system administrators were likely to be more restrictive, while those oriented toward developers were significantly more permissive.

6. The Relationship Between License Choice and Success

In Section 3 we showed that, holding the attractiveness of the project for the licensor constant, participation in an open source project should be negatively correlated with the restrictiveness of the license. To test this claim, we undertake a preliminary analysis of the relationship between license type and project success.

Table 10. Regression Analysis of Characteristics of Projects With and Without Highly Restrictive License Provisions, Comparing Early and Late Projects

	Dependent variable: all licenses highly restrictive?			
	Early projects only		Late projects only	
	Coefficient	Standard error	Coefficient	Standard error
Intended audience				
End-users/desktop	0.44	***0.09	0.36	***0.06
Developers	−0.40	***0.09	−0.16	***0.06
System administrators	0.05	0.10	0.14	**0.07
Topic				
Communications	0.13	0.11	−0.03	0.07
Security	−0.17	0.21	−0.10	0.13
Software development	−0.44	***0.11	−0.38	***0.07
Desktop environment	0.05	0.17	−0.13	0.14
Text editors	0.34	0.25	−0.20	0.13
Database	−0.16	0.15	−0.02	0.10
Education	0.23	0.24	−0.41	***0.13
Internet	−0.09	0.11	−0.07	0.07
Scientific/engineering	−0.13	0.14	−0.08	0.10
Multimedia	−0.15	0.11	−0.29	***0.09
Office/business	0.36	*0.20	−0.13	0.11
System tasks	−0.03	0.11	0.09	0.07
Printing	−0.18	0.49	−0.37	0.38
Terminals	−0.53	0.47	0.40	0.32
Games, etc.	0.20	*0.12	0.23	***0.08
Natural language				
French	0.18	0.15	0.14	0.11
Spanish	0.43	*0.26	0.22	0.15
Japanese	−0.33	0.38	−0.57	**0.24
German	0.17	0.15	0.15	*0.08
Russian	0.30	0.46	0.14	0.18
Environment				
X11	0.04	0.11	0.09	0.08
MS Windows	−0.20	0.13	−0.01	0.09
Other	−0.38	***0.11	−0.37	***0.07
Internet	0.08	0.11	0.002	0.07
No input/output	−0.28	**0.12	−0.29	***0.09
Cocoa (MacOS)	0.64	0.54	−0.12	0.16
Handhelds/PDAs			−0.14	0.21
Operating system				
POSIX	0.11	0.12	0.22	***0.07
Microsoft	−0.33	***0.11	−0.10	0.08
OS/2			0.38	0.65
MacOS	−0.27	0.19	−0.19	0.13
BeOS	−0.32	0.28	−0.62	**0.30
OS independent	−0.27	**0.11	−0.15	*0.07
PDA systems			0.25	0.29

Continued

Table 10. *Continued*

	Dependent variable: all licenses highly restrictive?			
	Early projects only		Late projects only	
	Coefficient	Standard error	Coefficient	Standard error
Development stage				
Pre-Alpha	−0.05	0.12	−0.12	*0.07
Alpha	−0.12	0.11	−0.09	0.07
Beta	−0.09	0.09	−0.001	0.07
Production/stable	−0.09	0.10	−0.18	**0.07
Mature	−0.05	0.20	−0.40	0.24
Constant	0.89	***0.16	0.64	***0.10
χ^2 statistic	278.36		362.18	
p-value	0.000		0.000	
Log-likelihood	−770.02		−1,761.45	
Number of observations	1,478		3,238	

The sample consists of 38,610 projects included in the SourceForge database in May 2002. The dependent variable is a dummy denoting whether all the licenses under which the project is licensed have highly restrictive provisions. The independent variables include dummy variables capturing various features of the open source projects. The first regression is restricted to those projects added to the SourceForge database before 2000; the second regression to those added in 2002. All regressions employ probit specifications.
*, **, *** Significant at the 10%, 5%, and 1% confidence levels, respectively.
In each regression, the following variables are omitted: the dummy variables denoting projects in the planning stage, those operating in a console (text) environment, those geared toward other audiences, those whose natural language is English, those geared toward another operating system, and those with another topic. Certain additional variables were dropped from the first regression due to collinearity.

To undertake this analysis, we first must identify a measure of project success. Because no single success measure is widely agreed upon, we employ a variety of metrics:

- The percentile ranking, based on activity since the inception of the project on SourceForge.
- The number of developers registered on SourceForge to work on the project (developers must sign up for individual projects to which they seek to contribute).
- The number of "bugs," or software errors, reported over the life of the project's stay on SourceForge. A dynamic program with frequent contributions and heavy use is likely to have many problems reported.[29]

As noted above, we restrict the sample to the approximately 10,000 projects with some activity on SourceForge.

Panel A of Table 12 presents the simplest analysis: a comparison of the activity of each project for each class of license. The results suggest that

29. One important limitation to this analysis is that different types of software may differ in their optimal number of lines of code. A given project may be quite successful, but still garner fewer contributions than a less successful project of a different type. While we partially address this concern by employing dummy variables for project type, these controls are unlikely to be perfect.

Table 11. License Type of Projects that are Corporate Spin-Offs

Regression	Coefficient	Standard error
All highly restrictive [7.1]	0.07	0.29
Some highly restrictive [7.2]	0.38	0.31
All restrictive [8.1]	0.32	0.34
Some restrictive [8.2]	0.46	0.37
Five-part index [9.1]	0.31	0.44
Three-part index [9.2]	0.20	0.44

The sample consists of 38,610 projects included in the SourceForge database in May 2002. The dependent variables are the six used in Tables 7–9, including dummy variables denoting whether the licenses had highly restrictive or restrictive provisions, as well as the two indexes of license type. The corresponding regression is denoted in brackets. In each case, the table reports the coefficient and standard error of a measure denoting whether the project was a corporate spin-off. Controls for the development stage, environment, intended audience, natural language, operating system, and topic are also employed, but not reported.

consistently greater contributions are made to projects that are not highly restrictive and (on a less consistently statistically significant basis) not restrictive. Our interpretation of this result, however, must be cautious. After all, we already know that quite different projects are chosen for each class of license.

We first address this concern by estimating ordinary least squares regressions, in which the dependent variables are the percentile activity ranking, the number of developers, and the number of reported bugs. We use the same independent variables as above, except that we also add one of the measures of license type. As Panel B of Table 12 reports, licenses with restrictive and highly restrictive features continue to be associated with fewer contributions.

We then explore the robustness of the results in several ways. We alter the specification (e.g., using the logarithm of the number of bug reports and developers). We also address concerns about sample selection bias in our sample: that is, the near certainty that some firms have zero activity not because they are based elsewhere, but rather because there really were no contributions. We address this concern by estimating the regressions including all observations, as well as by estimating two-stage “Heckit” regressions. The results continue to be of approximately the same magnitude and significance. At the same time, we believe that we can only partially control for the differences in project characteristics, so our interpretation of this preliminary analysis must be cautious.

7. Conclusion

This article examines the scope of licensing in open source software, a topic of both academic and practical interest. We first enumerate the various considerations that should figure into the licensor’s choice of contractual terms. We highlight how the decision is shaped not just by the preferences of the licensor itself, but also by that of the community of users. For instance, a commercial company releasing software to the open source community may choose a more restrictive license because of suspicion about its ultimate intentions.

Table 12. Measures of Activity in Open Source Projects, by License Type

Panel A: Univariate comparisons of activity

	Activity percentile	Number of developers	Bug reports
All highly restrictive			
Yes	50.8	2.7	9.3
No	***53.4	***3.5	**13.0
Some highly restrictive			
Yes	51.1	2.8	9.7
No	**53.4	***3.4	*12.5
All restrictive			
Yes	51.6	2.9	9.5
No	52.7	***3.6	**14.7
Some restrictive			
Yes	51.7	2.9	9.8
No	52.1	***3.5	**14.0

Panel B: Coefficients and standard errors from regression analyses

	Dependent variable					
	Activity percentile		Number of developers		Bug reports	
	Coefficient	Standard error	Coefficient	Standard error	Coefficient	Standard error
All highly restrictive	−2.39	***[0.92]	−0.59	***[0.12]	−4.21	**[1.72]
Some highly restrictive	−2.07	**[0.94]	−0.41	***[0.13]	−3.71	**[1.77]
All restrictive	−0.06	[1.04]	−0.59	***[0.14]	−3.57	*[1.95]
Some restrictive	0.13	[1.08]	−0.51	***[0.15]	−3.24	[2.02]

The sample consists of 9257 projects with some software contributions included in the SourceForge database in May 2002. The variables of interest are the overall activity percentile (with 100 being the highest) that the project fell into, the number of developers, and the total number of bugs reported. The significance level from a *t*-test is reported in each case where the null hypothesis of no difference is rejected. The second panel presents the coefficients and standard errors for the license type from ordinary least squares regression analyses with these three activity measures as independent variables. In each case, one of the license type measures is used as an independent variable, as well as controls for the intended audience, topic, natural language, environment, operating system, and development stage (not reported). Standard errors are in brackets.

*, **, *** Significant at the 10%, 5%, and 1% confidence levels, respectively.

The article then presents an empirical analysis of the prevalence and success of different types of open source licenses, employing the SourceForge database, a compilation of nearly 40,000 open source projects that has hitherto been largely unexplored by academics. The results are largely consistent with the framework in Section 3:

- Restrictive licenses are more common for applications geared toward end-users and system administrators, but significantly less common for those applications aimed toward software developers.

- Applications that are consumer oriented (e.g., desktop tools and games) are substantially more likely to have restrictive licenses. Those geared to the software development process are much less so.
- Projects whose natural language is not English, whose community appeal may be presumed to be much smaller, are more likely to employ restrictive licenses.
- Restrictive licenses are less common for projects operating in commercial environments or that run on proprietary operating systems.
- Projects with less restrictive licenses tend to attract more contributors.
- The results continue to be robust when we employ additional specifications and control variables, and when we examine only the projects posted early and late in the history of SourceForge.

This version of the article leaves a number of issues open, which we hope will be explored in subsequent work. In particular, two avenues seem promising ones for further study:

- The first of these is the consequence of the choice of license on project success. While we have taken a first look at these issues here, more should be done. One particularly promising empirical strategy is to examine projects that have changed from a restrictive to a permissive license (and vice versa). Ideally, looking at the reaction to license changes could be highly revealing. While the dataset used in this article is only a cross-sectional one, we have begun a new research project, which examines a panel of more then 100 open source projects and their contributors. This new dataset will allow us to undertake exactly this type of analysis.
- Second is the impact of open source license type on social welfare. It goes without saying that a license choice that is privately optimal from the point of view of the licensor may not be socially optimal. Some critics have argued (e.g., Mundie, 2001) that restrictive licenses are socially detrimental, because they reduce the incentives to innovate. These claims, not surprisingly, have been highly controversial. Obtaining a better understanding of these issues is an important challenge.

References

Dodd, Jeff C., and Brian Martin. 2000. "Building a Cathedral Over the Bazaar: A Preliminary View of Certain Licensing Practices in the Open Source and Free Software Communities," working paper, Mayor, Day, Caldwell & Keeton.

Gallini, Nancy T. 1984. "Deterrence by Market Sharing: A Strategic Incentive for Licensing," 74 *American Economic Review* 931–941.

Gallini, Nancy, and Brian D. Wright. 1990. "Technology Transfer Under Asymmetric Information," 21 *Rand Journal of Economics* 147–160.

Gandal, Neil, and Katharine Rockett. 1995. "Licensing a Sequence of Innovations," 47 *Economics Letters* 101–107.

Hammerly, Jim, Tom Paquin, and Susan Walton. 1999. "Freeing the Source: The Story of Mozilla," in Chris DiBona, Sam Ockman, and Mark Stone, eds., *Open Sources: Voices from the Open Source Revolution*. Cambridge, MA: O'Reilly.

Katz, Michael L., and Carl Shapiro. 1986. “How to License Intangible Property,” 101 *Quarterly Journal of Economics* 567–589.

Lee, Steve H. 1999. “Open Source Software Licensing,” working paper, Harvard University.

Lerner, Josh, and Jean Tirole. 2002. “Some Simple Economics of Open Source,” 52 *Journal of Industrial Economics* 197–234.

McGowan, David. 2001. “Legal Implications of Open-Source Software,” *University of Illinois Law Review* 241–304.

Mundie, Craig. 2001. “The Commercial Software Model,” available at http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.asp (accessed September 17, 2002).

Neukom, William H., and Robert W. Gomulkiewicz. 1993. “Licensing Rights to Computer Software,” in *Technology Licensing and Litigation 1993*. Practicing Law Institute Patents, Copyrights, Trademarks and Literary Property Course Handbook Series no. G4-3897. New York: Practicing Law Institute.

Perens, Bruce. 1999. “The Open Source Definition,” in Chris DiBona, Sam Ockman, and Mark Stone, eds., *Open Sources: Voices from the Open Source Revolution*. Cambridge, MA: O’Reilly.

Rockett, Katharine E. 1990. “Choosing the Competition and Patent Licensing,” 21 *Rand Journal of Economics* 161–172.

Shepard, Andrea. 1987. “Licensing to Enhance Demand for New Technologies,” 18 *Rand Journal of Economics* 360–368.

Williamson, Oliver. 1975. *Markets and Hierarchies: Analysis and Antitrust Implications*. New York: Free Press.

———. 1985. *The Economic Institutions of Capitalism*. New York: Free Press.