NBER WORKING PAPER SERIES

THE SCOPE OF OPEN SOURCE LICENSING

Josh Lerner
Jean Tirole

Working Paper 9363
http://www.nber.org/papers/w9363

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
December 2002

We thank Larry Augustin, Jeff Bates, and Pat McGovern for access to the SourceForge database. Comments by Yochai Benkler, Keith Bostic, Peter Childers, Jacques Crémer, David Genesove, Brian Kahin, Marten Miklos, Siobhan O'Mahony, Bruce Perens, Bernie Reddy, Larry Rosen, Marcin Strojwas, and participants in the conference "Open Source Software: Economics, Law and Policy" at the University of Toulouse and in 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 French Telecom and the Microsoft Corporation. All errors are our own. The views expressed herein are those of the authors and not necessarily those of the National Bureau of Economic Research.

The Scope of Open Source Licensing
Josh Lerner and Jean Tirole
NBER Working Paper No. 9363
December 2002
JEL No. O3, K3

ABSTRACT

This paper is an initial exploration of the determinants of open source license choice. It first enumerates the various considerations that should figure into the licensor's choice of contractual terms, in particular highlighting how the decision is shaped not just by the preferences of the licensor itself, but also by that of the community of developers. The paper then presents an empirical analysis of the determinants of license choice using the Source Forge 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 those geared towards the Internet are less likely to have restrictive licenses. Finally, projects that are likely to be attractive to consumers such as games are more likely to have restrictive licenses. A more tentative conclusion based on a much smaller sample is that projects that involve software developed in a corporate setting are likely to have more restrictive licenses. These findings are broadly consistent with theoretical predictions.

Josh Lerner
Harvard Business School
South Hall, Room 220
Boston, Massachusetts 02163
and NBER
jlerner@hbs.edu

Jean Tirole
Institut d'Economie Industrielle
Manufacture des Tabacs
Bureau MF529 - Bat. F
allees de Brienne
31000 Toulouse - FRANCE
tirole@cict.fr

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, and the nature of the fees that should be charged (e.g., the tradeoff 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 paper 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

[1]Gallini and Wright [1990] and Katz and Shapiro [1986] 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 (Gandall 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]).

distribute a modified version of the program as a proprietary commercial product without releasing the underlying source code.[2]

This paper 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, and restrictive ones commonplace when the appeal is more fragile. We suggest that projects geared towards developers may be more likely to fall into the former category, while those geared towards individual end users are more likely to fall into the latter.

The paper 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 paper, 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

[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.

clause is sometimes termed a "reciprocal" or a "viral" provision. For purposes of the empirical analysis in this paper, we term this a "highly restrictive" requirement.

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 (for example, the BSD license), restrictive (e.g., LGPL), and highly restrictive (GPL). (See below for a fuller discussion of these licenses.)

The results are largely consistent with the framework above: more restrictive licenses are more common in projects geared towards end users and in such applications as games and desktop applications. We explore the robustness of the results to the use of a variety of definitions of the independent variables. In an exploratory analysis using a much smaller sample, we examine the licensing terms of projects that are spun-out of corporations. The results are at least broadly consistent with theoretical suggestions.

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,

[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).

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 the GNU Public License, or GPL.[6] This license essentially required that the program's source code (the underlying programming commands) must 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

[4]Subsequently, software was also made available under formal contracts between developers and users. Later (in the personal computer era), software was protected through via 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 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.

[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).

software developers to undertake such activities in two critical ways: by insuring 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, or the 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[7]-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. 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 the LGPL.

[7]BSD stands for Berkeley Software Distribution. The credit provision was dropped in later versions of the license.

- 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 fee.
- 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 which 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

[8]For detailed analyses of the Open Source Definition, see Lee [1999] and Perens [1999].

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.

---

[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 product, 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 could 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 license, 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 into 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 requite 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).

3. The Choice of License: Some Considerations

A. Some Tradeoffs

Suppose that an individual or organization, whom we will term the "licensor," wants to start an open source project. The licensor may be a single developer, a group of developers with similar needs, or a corporation. As a first step, some initial code is written or gathered, and then released under some license. The choice of this license may be one of the key decisions of the overall design. The license—along with the quality of the released code, the licensor's reputation, and the demand for the product—will influence whether the project will appeal to programmers.

To be certain, the choice of a license may be affected by considerations that either lie outside standard utility-maximizing paradigms, or may be distorted by a misunderstanding of the implications of the alternative licenses in the choice set. Examples of the latter are hard to document in view of the short history of the open source movement, and any guess as to the current existence of such mistaken impressions is necessarily subject to debate. 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 license.[13] Our primary interest, though, lies in assessing the extent to which the initial choice of license is a rational choice.

[13]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.

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. The choice of a license impacts:

- The community of programmers who are asked to work on their project, as its benefits from working on the project may depend on the choice of license.[14]
- The end users, who may for example care about possible incompatibilities among versions or about the number of available applications. The choice of license, by affecting the likelihood of forking or the incentives of application developers, therefore impacts their welfare.
- The other open source projects that later will compete with or complement the project. For example, a GPL program may prove of no use for another open source project licensed under a BSD license that could otherwise have made use of the program.
- Commercial software vendors and support providers, whose opportunities are affected by the license.

When selecting a license, the licensor assesses the various benefits that the open source project will bring her. These include:

- The intrinsic motivation that the intellectual challenge provides.
- The signalling benefits (which encompass ego gratification and "career concerns" incentives such as future job offers and access to venture capital).
- The need to solve concrete problems for one's employer.

[14]As we will later emphasize, the externality cannot be too large since the licensor must secure the participation of the community, but this does not imply that the preferences of the licensor and the community are perfectly aligned.

- The possibility of material benefits.

For individuals, the latter includes the possible option of later building a commercial operation around the open source code. This material incentive is distinct from the career concerns incentives mentioned above. It depends crucially on the commercial reward being associated with an addition to the initial open source project. By way of contrast, many of the signalling benefits arise even if the subsequent work of the programmer is unrelated to the open source project. For corporations, material benefits include the increased profit on services or software that complements the open source software and the emancipation from the mark-ups and conditions imposed by a dominant software vendor with whom the open source project is meant to compete.

This mixture of motivations implies that the licensors have a wide variety of goals. For example, material benefits are paramount when licensors are corporations. Such benefits provide a smaller, but in a number of cases non-negligible, motivation in the case of individual licensors. The licensor must assess how her mixture of motivations, together with project characteristics—such as the environment, the size of the initial code base, and the intended audience—impacts the following general considerations:

1) Interaction of project with other software.

As mentioned above, the open source project under consideration 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.

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 (see below), 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 standard setting 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 the 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. The "greenfield" considerations discussed shortly thus need to be augmented by an analysis of "legacy aspects."[15]

[15]An additional complication is introduced by the asymmetry of the licenses, especially the greater restrictions in the GPL license. 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-licensed projects,

2) Hijacking.

Advocates of restrictive licenses argue that unrestrictive ones 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. While the resulting software may (or may not) be superior, the firm disrupts the dynamics of the open source project by *de facto* privatizing it. (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. (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 our 2002 paper—and that these network effects may be reduced by competition from a proprietary variant.) This prospect may discourage potential contributors in the first place.

This argument for restrictive licenses could be rephrased as saying that community members make project-specific investments. Hijacking poses the

---

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.

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 "hold up" problem.

To be certain, restrictive licenses are not immune to a problem akin to hijacking themselves. After all, commercial software vendors can rewrite the open source code. (Copyright protection of software—unlike patent protection—only protects the expression, not the fundamental ideas.)

In the end, the risk of hijacking under alternative licenses depends on the nature of the project. Open source projects that are conservative reimplementations of pre-existing software are probably less subject to hijacking than innovative software products.[16] Another potential determinant is the size of the code. Large projects are more costly to rewrite, and so costs and delay factors may make the choice of license more relevant in this case.

3) *Impact of software patents*.

Open source software proponents have often expressed concerns that patent infringement suits may hamper the projects. It is easy to imagine circumstances under

[16]Bezroukov [2002] puts Linux in the former category, and scripting languages (TCL, Perl, Python, PHP) in the latter.

which software patents might affect the dynamics of the process. Contributors to the software, as well as users and distributors of the code, may be deterred, especially corporations. The GPL specifies that if some contribution is found to be infringing on a patent and the ability to distribute the code (or modification of the code) is thereby restricted, then the code cannot be distributed at all. This covenant is meant to prevent "joint hijacking" by the patent owner and an open source infringer who would then receive an exclusive license from the patent owner. Historically, these provisions have not been included in any of the non-GPL contracts.

4) *Impact on incentives to produce complementary software*.

A standard argument in favor of unrestrictive licenses is that permissiveness is what it takes to attract commercial software developers to write applications that enhance the value of the open source code. In particular, it has been suggested that in mature projects, when the energy of the initial contributors may be fading, the involvement of commercial contributors may be critical to success (Bezroukov [2002]). (We will discuss license choices in cases where firms open up proprietary software below.)

5) *Familiarity of open source community with the license*.

There are benefits in the form of reduced transaction cost to the licensor who adopts a familiar license rather than an innovative but unfamiliar one. Licensors choosing a well-known license economize on the learning costs incurred by the community as to how the license works and what its likely implications for the development process are.

6) *Forking*.

Forking refers to an internal threat of competing groups moving in different directions and producing incompatible versions of the same initial open source project. It is unclear to us how license type will affect the probability of forking or the effectiveness of the original project leader's response; this topic may reward future research.

B. Enlisting the Developers

We now consider some of these issues more formally. Consider a licensor (an individual, group of individuals, or a corporation) choosing among K different licenses, $k=1,...,K$. License k confers expected payoffs or utilities U_L^k and U_C^k for the licensor and the (representative member of the) developer community, respectively.

For example, these payoffs may take the following form (for i equal to L or C):

$$U_i^k = B_i^k + \pi_i^k,$$

where

B_i^k is the sum of the signalling benefit (peer recognition, career concerns) and the potential benefit of being able to tailor the code for one's specific usage (B_i^k may also include the pleasure of working in a type-k open source environment),

and

π_i^k is the expected commercial incentive. For an individual, this would include the option of providing services or services based on the open source project, perhaps through a start-up. For the corporate licensor, this would include the

option of privatizing the code later on, an increase in the sale of a complementary proprietary software due to the development of the open source project, or the reduction of the mark-up of another commercial software due to competitive pressure of the open source program.

Letting $\overline{U}_C$ denote the opportunity cost of participating in the open source project of the (representative member of the) community,[17] then the choice of license is governed by a constrained optimization. All else being equal, the licensor would like to choose her preferred license, but must satisfy the open source community's participation constraint (CPC):

$$\max_{\{k\}} U_L^k$$

$$\text{s.t.} \qquad U_C^k \geq \overline{U}_C. \qquad (CPC)$$

Furthermore, it must be the case that the resulting choice satisfy the "licensor's participation constraint" (LPC). Let $\overline{U}_L$ denote the payoff to the licensor of keeping the code private rather than releasing it under an open source license (the licensor may undertake the project as a proprietary project, or may just work on alternative projects). Then it must satisfy the constraint:

$$U_L^k \geq \overline{U}_L. \qquad (LPC)$$

Let us for simplicity assume that there are only two types of licenses—restrictive (*R*) and permissive (*P*)—and make the following assumption:

[17]The member could alternatively work on other projects (commercial or open source).

If $U_L^R \geq U_L^P$, then *a fortiori* $U_C^R \geq U_C^P$.

The motivation for the assumption that the project leadership (the licensor) is relatively more likely to benefit from a permissive license is that the ability to demonstrate talent to one's peers and/or to the labor market is not much affected by the choice of license. But commercial benefits, which are probably larger under a permissive license, 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 such opportunities. Put another way, because the leadership of the project is likely to benefit more than community from a permissive license, if a restrictive license is better for the leadership then one can assume that such a license will also be better for the community. 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.[18]

Ignoring for the moment the licensor's participation constraint, we can distinguish two cases:

- *Strong community appeal*: The community will participate if the licensor wants to opt for the permissive license $\left(U_C^P \geq \overline{U}_C\right)$. In this case, the licensor chooses between the restrictive and permissive licenses in an unconstrained fashion.

[18]The logic behind this reasoning may be less compelling in the case where a corporation makes available proprietary software that it is already developed under an open source license. We discuss this special case below.

- *Fragile community appeal*: The licensor will not obtain participation if she opts for the permissive license $\left(U_C^P < \overline{U}_C \leq U_C^R\right)$. Thus, the licensor must opt for the restrictive license, whether the latter is her unconstrained preferred choice or not.

To illustrate the impact of community participation on licensing choice, suppose that the community of developers expects no financial reward from being able to commercialize complementary proprietary software or support $\left(\pi_C^R = \pi_C^P = 0\right)$. Suppose further that their benefits from ego gratification, career concerns, and open source interactions satisfy the condition that:

$B_C^P < \overline{U_C} < B_C^R$.

This setting is depicted in Figure 1. For convenience, the figure normalizes the licensor's benefits $B_L^k\left(k = P, R\right)$ to be equal to those of the community, although in practice they may differ (e.g., the leadership may get greater benefits). The line S_0 represents an indifference curve, denoting combinations of benefits that provide equal levels of satisfaction to the licensor. If the choice is between points 1 and 2, the leadership prefers the restrictive license: the higher financial prospects from a permissive license are not sufficiently large to make it appealing to the licensor. The restrictive license is then a Pareto choice to the extent that both parties prefer it. By way of contrast, if the choice were between points 2 and 3, the licensor would prefer the permissive license, but instead chooses the restrictive one so as to enlist other programmers.

One application of this framework concerns projects with unsophisticated end users as the intended audience, such as desktop applications and games. It is plausible to regard these as part of the "fragile community appeal" category:

- 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.
- The benefits from tailoring the code for particular applications are weak.

By way of contrast, 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 towards unsophisticated end users.[19]

One interesting question relates to the licenses chosen by corporations when they release code. It might be thought at first glance that corporations would universally employ permissive licenses, since they wish to retain the right to commercially exploit discoveries. But we should actually not be surprised if we see firms choosing more restrictive licences. The very fact that the licensor could keep the code private (the licensor's participation constraint) implies that the act of releasing the code creates a "truncation effect." Corporations release some pieces of code because their chance of winning the commercial battle against a rival has become small (say, due to technological differences or network externalities). As a result, the corporation prefers gambling on a

[19]In the analysis below, we will equate certain licenses (denoted in Table 1) with the restrictive licenses discussed here. This need not be the case for our analysis to hold: these licenses need only be *perceived* as more restrictive. The latter assumption seems abundantly justified (e.g., Bezroukov [2002], Dodd and Martin [2000], Lee [1999]).

different strategy, such as selling consulting services or licensing the code on a case-by-case basis to customers that for some reason cannot make use of the product if subject to a restrictive license.[20] Such code may therefore belong to the fragile community appeal category, and thereby be more likely to receive a restrictive license.

A related, but different, point is that the choice of initiating an open source project may be subject to an adverse selection problem if the community is less well informed than the licensor. The community may be suspicious about the project's prospects (the licensor may have released the code because the commercial prospects were low) or about the licensor's intent (such as its commitment to the project rather than to commercially adjacent segments and the possibility that the firm will reprivatize the project). This latter concern about reprivatization may induce the licensor to choose a restrictive license in order to "prove" her good faith.[21]

4. 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 service that since 1999 has offered hosting and project administration tools to software development

[20]MySQL AB follows the latter strategy with its MySQL database. The firm simultaneously sells its software and makes it available for free under the GPL. Many large corporations prefer to purchase their product, both because of liability concerns (the GPL product is made available on an "as is" basis, while purchasers of the commercial product are fully indemnified) and worries about the GPL.

[21]The case of Netscape's Mozilla project, whose initial license was greeted with protests and was replaced by a more restrictive license, is one illustration (Hammerly, *et al*. [1999]).

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 computer systems optimized for Linux. Today, VA Software 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 thousand 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, however, to be much smaller: Savannah, often referred to as SourceForge's leading competitor, had 790 active projects in May 2002.[23] 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 was 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.

---

[22]There are, however, exceptions. 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]http://savannah.gnu.org (accessed September 17, 2002).

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 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—e.g., 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 thousand 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 final columns of Table 1 and Table 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 General Public License is clear. Fully 72% of the licenses are the GPL, 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 to end-users and developers, and geared to two families of operating systems (the POSIX

[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 the 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 suggests that the GPL license is not as dominant: 63% of the weighted projects have GPL licenses, 11% have Lesser GPL licenses, and 11% BSD licenses.

family—which includes Linux, BSD, and Sun's Solaris—and Microsoft) or else independent of any operating system.

A natural concern is the extent to which the measures are co-linear: in other words, the extent to which the characteristics of the projects are highly correlated with each other. Table 3 provides an illustrative tabulation displaying the cross-tabulation of project topic and intended audience. To be sure, there is some clustering: for instance, projects geared towards 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 the other comparisons.

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 4 highlights several patterns:

- Highly restrictive licenses are less common for more mature projects. This pattern may reflect a "vintage effect": it may have more common for older projects to employ licenses other than the GPL. Alternatively, this may reflect a "survival effect." Projects with the GPL may have been less successful in attracting contributions. (When we examine the impact of these factors on different "vintages" of projects below, we will be able to shed some light on this question.)
- 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.
- Highly restrictive licenses are significantly more common for projects that run under the POSIX family of operating systems, as opposed to other proprietary ones (or those which are operating system independent).
- Consistent with the framework in Section 3.B, highly restrictive licenses are more common for applications geared towards end-users, but significantly less common for those applications aimed towards software developers. Highly restrictive

licenses are also more common for projects geared to systems administrators, which may reflect either the weak community appeal of these efforts or the intrinsic preferences of the licensors (since commercial benefits are likely to be low).

- 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 much more common for projects whose natural language is other than English, with the exception of Japanese.

When we examine in Table 5 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 6 and 7 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 towards other audiences,

those whose natural language is English, those geared toward an other operating system, and those with an other topic.

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

- Software geared toward developers is sharply different from that geared towards 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. The tendency to see fewer such licenses in Internet-related projects is consistent with the arguments concerning standard setting 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 be so.

The results in Table 7 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 6. 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 towards individual end-

users has a 23% higher probability of all licenses being highly restrictive than one oriented to developers.

The regression analysis in Table 8 looks at restrictive and highly restrictive licenses in a single specification. To do this, we employ indexes, which 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 having a narrower scope than one rated as a "2," but not necessarily twice as much so. The findings in Table 8 are largely consistent with the analyses reported above, particularly those in Table 7.

One concern with the analysis in Table 8 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 re-estimate 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 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 9 shows the most direct approach. We re-estimate the regression reported in the first column of Table 6, 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 do so. Projects that are designed to run on

commercial operating systems are less likely to have highly restrictive licenses. Finally, types of projects that are likely to be attractive to consumers—such as games—are more likely to have highly restrictive licenses.

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 data 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.B 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 10 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 a manner that introduces some biases. We rerun the regressions reported here, restricting the sample to the approximately ten thousand observations with code contributions. We also repeat the analysis, weighting the observations by a number of activity measures: the numbers 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 discussed in Section 3.A may distort the decisions being made. To partially address this concern, we re-ran the regressions reported in Table 8, eliminating those with BSD and GPL licenses, 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. Conclusions

This paper 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.

The paper 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 above:

- Restrictive licenses are less common for projects operating in commercial environments or that run on proprietary operating systems.
- Consistent with the framework in Section 3.B, restrictive licenses are more common for applications geared towards end-users and system administrators, but significantly less common for those applications aimed towards software developers.
- Also consistent with the framework, 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.
- Similarly, products geared to technical users are less likely to have restrictive licenses.

This version of the paper 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 getting a better understanding of the other key inputs that go into the choice of license. For instance, how does the fear of adverse outcomes such as hijacking, forking, and the failure to develop complementary software products change with the type of project, its stage of development, and the nature of the licensor? How do the license terms of complementary software products impact with the choice of license?
- Second, the consequence of the choice of license on project success is an interesting issue. To what extent does this decision matter? It might be possible to identify cases where licensors were constrained in their choice of license, which might allow the implications of license type to be identified.

References

Bezroukov, Nikolai, 2002, "BSD vs. GPL, Part 2: The Dynamic Properties of BSD and GPL Licenses in the Context of the Program Life Cycle," http://www.softpanorama.org/Copyright/License_classification/social_dynamics_of_BSD_and_GPL.shtml (accessed September 17, 2002).

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," Unpublished working paper, Mayor, Day, Caldwell & Keeton.

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

Gallini, Nancy, and Brian D. Wright, 1990, Technology Transfer under Asymmetric Information," *Rand Journal of Economics*, 21, 147-60.

Gandal, Neil, and Katharine Rockett, 1995, "Licensing a Sequence of Innovations," *Economics Letters*, 47, 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, editors, *Open Sources: Voices from the Open Source Revolution*, Cambridge, Massachusetts, O'Reilly, pp. 197-206.

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

Lee, Steve H., 1999, "Open Source Software Licensing," Unpublished working paper, Harvard University.

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

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

Mundie, Craig, 2001, "The Commercial Software Model," 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, 1993), New York, Practicing Law Institute, pp. 775-___.

Perens, Bruce, 1999, "The Open Source Definition," in Chris DiBona, Sam Ockman, and Mark Stone, editors, *Open Sources: Voices from the Open Source Revolution*, Cambridge, Massachusetts, O'Reilly, pp. 171-188.

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

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

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

Williamson, Oliver, 1985, *The Economic Institutions of Capitalism*, New York, The Free Press.

Figure 1: Illustration of license choice.

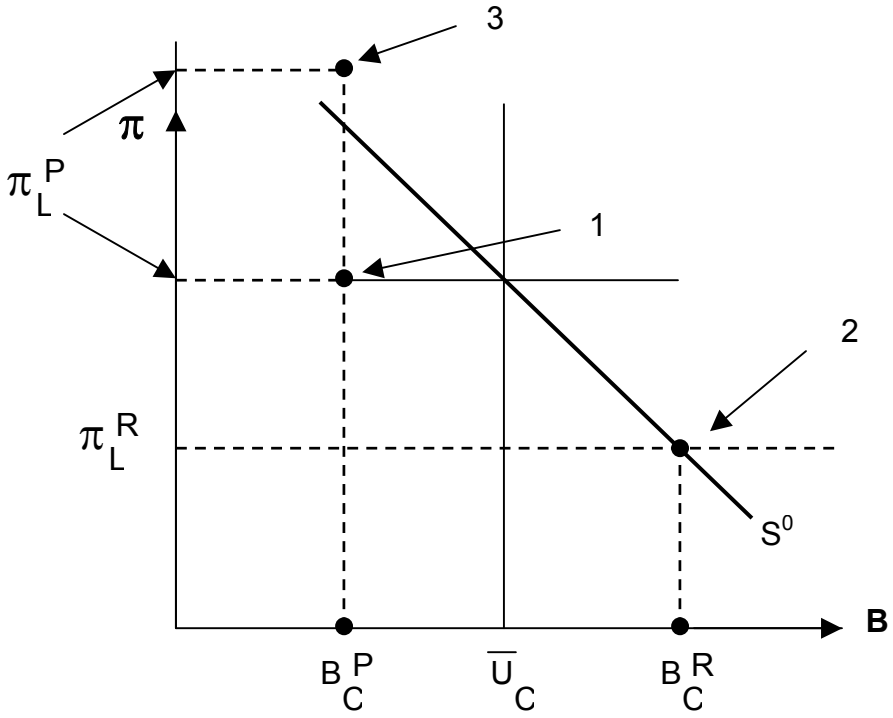

Table 1: Open source software licenses. The table summarizes all Open Source Initiative-approved licenses, as well as selected others. The final two columns indicate the number of observations of each license type in the SourceForge database.

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	5,801
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	1,047
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

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.

Abbreviations:
L = License
OS = Open Source
PL = Public License

Notes:
[a]Licensees can choose between two possible options.
[b]Deviations from certain industry standards, however, must be documented.

Table 2: Characteristics of the SourceForge sample. The table summarizes 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 for the subset of 9,257 observations where SourceForge has data on software contributions. Some projects may be classified into multiple categories.

Panel A: Percentage Distribution of Entire Sample

Development Stage		*Environment*		*Intended Audience*		*Natural Language*		*Operating System*		*Topic*	
Planning	31.8	Console (Text)	30.7	End Users/Desktop	55.8	English	95.8	POSIX	57.1	Communications	16.2
Pre-Alpha	21.3	X11	29.0	Developers	64.1	French	5.7	Microsoft	28.6	Security	3.2
Alpha	18.9	MS Windows	24.1	System Administrators	24.5	Spanish	3.1	OS/2	0.2	Software Dvlpmt.	17.6
Beta	22.0	Other	13.5	Other	14.2	Japanese	1.1	MacOS	3.5	Desktop Environ.	4.5
Production/Stable	16.8	Internet	31.4			German	9.2	BeOS	0.9	Text Editors	2.8
Mature	1.8	No Input/Output	10.3			Russian	1.3	OS Independent	36.8	Database	6.9
		Cocoa (MacOS)	1.0					Other	2.0	Education	3.2
		Handhelds/PDAs	0.3					PDA Systems	0.1	Internet	24.0
										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

Development Stage		*Environment*		*Intended Audience*		*Natural Language*		*Operating System*		*Topic*	
Planning	8.1	Console (Text)	32.7	End Users/Desktop	53.1	English	96.7	POSIX	57.9	Communications	15.3
Pre-Alpha	12.4	X11	30.2	Developers	66.5	French	5.4	Microsoft	29.8	Security	3.6
Alpha	22.0	MS Windows	25.1	System Administrators	28.5	Spanish	2.9	OS/2	0.3	Software Dvlpmt.	21.2
Beta	36.2	Other	13.1	Other	12.6	Japanese	1.2	MacOS	3.7	Desktop Environ.	5.1
Production/Stable	32.0	Internet	27.9			German	8.7	BeOS	1.0	Text Editors	3.4
Mature	3.3	No Input/Output	10.6			Russian	1.7	OS Independent	36.4	Database	7.1
		Cocoa (MacOS)	1.2					Other	1.9	Education	2.7
		Handhelds/PDAs	0.3					PDA Systems	0.2	Internet	24.1
										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

Table 3: Cross-tabulation of intended audience and project topic. 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.

	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%

Table 4: Tabulation of characteristics of projects with and without highly restrictive license provisions. The sample consists of 38,610 projects included in the SourceForge database in May 2002. The table summarizes 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 χ^2-test is reported in each case where the null hypothesis of no difference is rejected.

	All Licenses Highly Restrictive?		*Some Licenses Highly Restrictive?*	
	Yes	No	Yes	No
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
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
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
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
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
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
Topic				
Communications	17.0	***14.1	16.9	***14.1
Security	3.2	3.1	3.3	2.9
Software Dvlpmt.	12.2	***29.6	12.8	***30.3
Desktop Environ.	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/Enging.	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, et al.	16.6	***12.3	16.5	***12.1

Definitions:
* = Significant at the 10% confidence level.
** = Significant at the 5% confidence level.
*** = Significant at the 1% confidence level.

Table 5: Tabulation of characteristics of projects with and without restrictive license provisions. 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 χ^2-test is reported in each case where the null hypothesis of no difference is rejected.

	All Licenses Restrictive?		*Some Licenses Restrictive?*	
	Yes	No	Yes	No
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
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
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
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
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
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
Topic				
Communications	16.3	*15.2	16.3	15.2
Security	3.1	**3.7	3.1	3.4
Software Dvlpmt.	15.8	***25.5	16.0	***25.5
Desktop Environ.	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/Enging.	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, et al.	15.9	***12.7	15.8	***12.7

Definitions:
* = Significant at the 10% confidence level.
** = Significant at the 5% confidence level.
*** = Significant at the 1% confidence level.

Table 6: Regression analysis of characteristics of projects with and without highly restrictive license provisions. 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.

	Dependent Variable:			
	All Licenses Highly Restrictive?		*Some Licenses Highly Restrictive?*	
	Coefficient	Standard Error	Coefficient	Standard Error
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
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
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
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
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
PDA Systems	0.23	0.26	0.33	0.26
Topic				
Communications	-0.01	0.03	-0.003	0.03
Security	-0.08	0.06	-0.06	0.06
Software Dvlpmt.	-0.40	***0.03	-0.36	***0.03
Desktop Environ.	-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/Enging.	-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, et al.	0.05	0.04	0.07	*0.04
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	

Definitions:
* = Significant at the 10% confidence level.
** = Significant at the 5% confidence level.
*** = Significant at the 1% confidence level.

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 towards other audiences, those whose natural language is English, those geared toward an other operating system, and those with an other topic.

Table 7: Regression analysis of characteristics of projects with and without restrictive license provisions. 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.

	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
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
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
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
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
Topic				
Communications	0.03	0.04	0.03	0.04
Security	-0.11	*0.06	-0.04	0.07
Software Dvlpmt.	-0.20	***0.04	-0.15	***0.04
Desktop Environ.	-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/Enging.	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, et al.	0.06	0.04	0.09	**0.04
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	

Definitions:
* = Significant at the 10% confidence level.
** = Significant at the 5% confidence level.
*** = Significant at the 1% confidence level.

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 towards other audiences, those whose natural language is English, those geared toward an other operating system, and those with an other topic.

Table 8: Regression analysis of characteristics of projects with and without various license provisions. The sample consists of 38,610 projects included in the SourceForge database in May 2002. The dependent variable indexes denoting 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 ordered logit specifications.

	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
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
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
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
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
Topic				
Communications	-0.003	0.05	0.0004	0.05
Security	-0.13	0.10	-0.17	0.10
Software Dvlpmt.	-0.51	***0.05	-0.52	***0.05
Desktop Environ.	-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/Enging.	-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, et al.	0.12	*0.06	0.10	*0.06
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	

Definitions:
* = Significant at the 10% confidence level.
** = Significant at the 5% confidence level.
*** = Significant at the 1% confidence level.

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 towards other audiences, those whose natural language is English, those geared toward an other operating system, and those with an other topic.

Table 9: Regression analysis of characteristics of projects with and without highly restrictive license provisions, comparing early and late projects. 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.

	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
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
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
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
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
Topic				
Communications	0.13	0.11	-0.03	0.07
Security	-0.17	0.21	-0.10	0.13
Software Dvlpmt.	-0.44	***0.11	-0.38	***0.07
Desktop Environ.	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/Enging.	-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, et al.	0.20	*0.12	0.23	***0.08
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	

Definitions:
* = Significant at the 10% confidence level.
** = Significant at the 5% confidence level.
*** = Significant at the 1% confidence level.

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 towards other audiences, those whose natural language is English, those geared toward an other operating system, and those with an other topic. Certain additional variables were dropped from the first regression due to collinearity.

Table 10: License type of projects that are corporate spin-offs. The sample consists of 38,610 projects included in the SourceForge database in May 2002. The dependent variables are the six used in Tables 6 though 8, 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 spinout. Controls for the development stage, environment, intended audience, natural language, operating system and topic are also employed, but not reported.

Dummy Denoting Corporate Spin-Off Projects		
Regression	*Coefficient*	*Standard Error*
All Highly Restrictive [6.1]	0.07	0.29
Some Highly Restrictive [6.2]	0.38	0.31
All Restrictive [7.1]	0.32	0.34
Some Restrictive [7.2]	0.46	0.37
Five-Part Index [8.1]	0.31	0.44
Three-Part Index [8.2]	0.20	0.44