Open Source Software: Motivation and Restrictive Licensing[#]

Chaim Fershtman and Neil Gandal[*]

April 11, 2007

Abstract

Open source software (OSS) is an economic paradox. Development of open source software is often done by unpaid volunteers and the "source code" is typically freely available. Surveys suggest that status, signaling, and intrinsic motivations play an important role in inducing developers to invest effort. Contribution to an OSS project is rewarded by adding one's name to the list of contributors which is publicly observable. Such incentives imply that programmers may have little incentive to contribute beyond the threshold level required for being listed as a contributor. Using a unique data set we empirically examine this hypothesis. We find that the output per contributor in open source projects is much higher when licenses are less restrictive and more commercially oriented. These results indeed suggest a status, signaling, or intrinsic motivation for participation in OSS projects with restrictive licenses.

Keywords: Open Source Software, Intrinsic Motivation, Professional Status, Signaling, Restrictive Licenses.

JEL Numbers: D20, L86

[#] We are grateful to an anonymous referee, the editors, Guenter Knieps and Ingo Vogelsang, and to Judith Chevalier, David Evans, Bernard Reddy, David Steinberg, Manuel Trajtenberg, and seminars participants at the 2004 AEA meetings and Tel Aviv University for helpful comments. Financial support from NERA is gratefully acknowledged.

[*] Fershtman: Department of Economics, Tel Aviv University, Tel Aviv, 69978, ISRAEL, Erasmus University and CEPR, fersht@post.tau.ac.il, Gandal: Department of Public Policy, Tel Aviv University, Tel Aviv, 69978, ISRAEL and CEPR, gandal@post.tau.ac.il.

1. Introduction

The open source model is a form of software development with source code that is often made publicly available and free of charge. Interested parties and users have the right to modify and extend the program.[1] The open source model has become quite popular and often referred to as a movement with an ideology and enthusiastic supporters.[2] At the core of this process are two interesting aspects. The first is that unpaid volunteers do a significant portion of the development of open source programs and the second is that many of the open source programs employ restrictive licenses.[3]

Having unpaid volunteers is puzzling for economists. What are the incentives that drive developers to invest time and effort in developing these open source programs? There are several possible explanations: Lerner and Tirole (2002) argue that developers of open source programs acquire a reputation, which is eventually rewarded in the job market, while Harhoff, Henkel and von Hippel (2003) argue that end users of open source benefit by sharing their innovations.[4] Johnston (2002) develops a model of open source software as voluntary provision of a public good – but for such a model one need to assume that the primary motivation of developers is the "consumption" or the use of the final program.[5]

The general view, however, is that personal satisfaction, ideology, and professional status may be important enough to provide incentives to software

[1] Open source is different than "freeware" or "shareware." Such software products are often available free of charge, but the source code is not distributed with the program and the user has no right to modify the program. Like other products based on intellectual property, the intellectual property in software is typically "licensed" for use, not sold outright. Someone who purchases a music CD buys the physical CD and the right to play the music under specific circumstances (which do not include the right to play it on the radio, etc). Software, whether proprietary or open-source, is similarly "licensed for use."

[2] See for example Raymond (2000) and Stallman (1999).

[3] The open source model has changed somewhat recently and now some of the work is performed by employees of companies who support open source software. While it is true that even in 2001-2002, a non-trivial number of open source project programmers were being paid for their contributions, Ghosh et al (2002) conclude that open source development was more like a hobby than a (paying) job. Our analysis, which was conducted using data from 2001-2002, focuses on the less well-known projects that were more likely to rely on volunteers.

[4] Hann, Roberts, and Slaughter (2002) examine the Apache HTTP Server Project and find that contributions are not correlated with higher wages, but a higher ranking within the Apache Project is indeed positively correlated with higher wages. But such a correlation will occur whenever a higher ranking reflects higher productive capabilities of programmers.

[5] Johnson (2005) present a model in which the OSS organization structure is superior to that of proprietary development as it minimizes transaction costs and avoids agency problems.

developers. Using a Web-based survey Lakhani and Wolf (2005) found that intrinsic motivations help induce developers to contribute to OSS.[6]

The important role of intrinsic motivation may suggest an analogy between academia and the open source movement. While publication plays an important role in academia the analogy in the OSS world is being included in the "list of contributors" of different projects. This formal recognition is likely viewed as an important verifiable reward for developers. Academic publication, however, typically provides much more detailed information about the contribution of authors. There are relatively fewer authors for each paper while in an OSS project there is typically a long list of "coauthors".[7] The difference in format likely affects the incentive of each "coauthor." A coauthor of an OSS project is a programmer whose contribution is deemed sufficient to be included in the list of contributors. Typically, there is a threshold level, which varies among programs. If the contribution is beyond the threshold level, the individual is rewarded by having his name appear in the list of contributors.[8] It is the inclusion in this list which provides the professional status and recognition.

If the objective of a programmer is to have his name included in the list of contributors, we expect that he will contribute up to the threshold level, but not much beyond it. This setting is probably similar to incentives to donate money to universities, museum, theaters etc. The list of benefactors is published and there is a threshold level above which a donation is recognized formally by being included in list of benefactors. While there are typically different levels of benefactors, each with different threshold levels, we would expect that most contributions are made at or slightly above the threshold levels. Similarly, whenever the main concern of programmers is to be included in the "list of contributors," we expect that their contribution will be at the threshold level and not much beyond it.

[6] See also Hars and Ou (2001), Hertel, Niedner, and Herrmann (2002). Using survey methods, these papers respectively find that peer recognition and identification with the goals of the project are the main motivations for developers who contribute to open source software projects.
[7] The mean number of contributors is 83 in our sample.
[8] In the XFree86 project, for example, in order to become a developer, one must submit an accepted patch (see Halloran and Scherlis (2002)).

Our paper exploits this property and empirically examines the output per contributor in different types of OSS. If our hypothesis is valid then we would expect that the output per contributor would be smaller in less commercial OSS projects.[9]

Open source software is often licensed under the General Public License (GPL). Any program that incorporates software from an open source program under a GPL must distribute the changes as open source software as well, also under a GPL. Hence the GPL is typically considered to be a relatively restrictive license. Other open source software programs, however, are distributed with Berkeley Software Distribution type (BSD-type) licenses. Commercial products can be developed using software licensed under a BSD-type license as long as credit is given to the organization responsible for the underlying code. Hence BSD-type licenses are considered less restrictive than the GPL and such programs can more easily be used for commercial purposes.[10,11] Previous work indeed suggests that less restrictive licensing is associated with more commercial use (see Lerner and Tirole (2005)). When developers have commercial concerns, the existence of a recognition threshold is probably less important in determining effort. Hence our hypothesis is that other things being equal, the average contribution per participant will be larger for projects employing non-restrictive licenses.

We use a unique panel data set of 71 open source projects over an eighteen-month period in 2002-2003. We examine how the type of license and other factors affect output per contributor in open source projects. Our data set is unique because source lines of code (SLOC) are available for the open source projects. For our purposes, SLOC is a good measure because we want to examine the effort that is put into the project, rather than whether a project succeeds.

Our main result is that the output per contributor in open source projects is much higher when licenses are less restrictive. The strong correlation in the data between output per contributor and license type remains even after we control for all

[9] Most of the empirical work in the literature has focused on case studies of individual well-known open source projects such as Linux and Apache. Hann et al (2002) study the Apache web server, while Kuan (2002) examines the Apache web server, the Linux operating system, and the Gnome user interface. However, as Shah (2002) noted there may be potential differences between a relatively small group of well-known open source projects and the general population of open source projects. The projects in our sample do not include the well-known successful open source projects, such as Apache, Bind, Linux, Perl and Sendmail which are not hosted at SourceForge.

[10] Nevertheless, BSD-type licenses are not like commercial licenses, since commercial firms do not typically provide the source code to users, nor do they let users redistribute the code.

[11] The definition of the GPL as a relatively restrictive license follows Lerner and Tirole (2005).

other factors that might affect output per contributor. Hence although we choose to estimate a particular specification, the main result that the mean output per contributor is significantly greater for projects that use non-restrictive licenses is completely robust to alternative specifications: there is a very robust empirical regularity. This result is striking since the type of license does not "technically" affect the writing of the code.[12]

Our result is thus consistent with the hypothesis that being included in the list of contributors is the key source of motivation for participation in open source projects with restrictive licenses. Potential contributors in projects with restrictive licenses (limited commercial potential) have strong incentives to contribute up to the minimum threshold level in order to be included in the "list of contributors." Once someone is included in the list of contributors, the incentive to contribute beyond that level is diminished considerably in such projects.

2. Type of License, Output, and Participation

Even though open source software is distributed freely without payment, the programs are distributed under licensing agreements. There are several different types of open source licenses. The main difference is the degree of restrictions they provide. We distinguish between three levels of (relative) license restrictiveness; very restrictive, moderately restrictive and non-restrictive (see Lerner and Tirole (2005) for further discussion).

- The most popular open source license is the GNU General Public License or GPL. This license requires that the programming commands (or source code) be made available to anyone and that other programs that incorporate code from a GPL licensed program must also make the source code fully available under the GPL. Hence, programs that use code from a GPL licensed program cannot become proprietary software. The GPL will be classified as a very restrictive license.

[12] It would, of course, be interesting to know whether a small number of individuals made most of the contributions, but such information is not available for our data set. While the distribution about this average would add some additional information, the results using averages are quite striking, and they enable us to answer our primary research question.

- Another popular license is the GNU Lesser GPL (or LGPL). This type of license is also quite restrictive, but less restrictive than the GPL restrictive license. We will refer to this as a "moderately restrictive" license. Most of the moderately restrictive licenses in our data set use the LGPL. Other moderately restrictive licenses in our data set include the Mozilla, MPL, and NPL licenses.
- The main alternative to the licenses described above is the Berkeley Software Distribution (BSD) type license which has fewer restrictions than the GPL and LGPL licenses. For example, commercial products can be developed using software licensed under a BSD-type license as long as credit for the underlying code is given to the copyright holder(s). Hence, we refer to the BSD-type licenses as "nonrestrictive" licenses. Other licenses in this category that appear in our data set include an MIT license, a Sun Industry Standards Source License, an Intel OSL and an Apache Software License.

The three types of licenses may provide different incentives for developers to participate in open source projects and to exert effort. Lerner and Tirole (2005) examine the choice of licenses using a very large database of open source projects from the SourceForge web site. They find that open source projects that run on commercial operating systems and projects that are designed for developers tend to use less restrictive licenses, while projects that are targeted for end users tend to use more restrictive licenses. From the firms' perspective, Bonaccorsi and Rossi (2002) surveyed Italian firms that use open source software and found that, on average, firms that employ software with restrictive licenses supply fewer proprietary products than firms that employ software with less restrictive licenses.

Our focus, however, is not on the choice of license but rather on how the choice of license type affects developers' incentives and consequently, participation and effort.

3. Data

We employ a unique data set consisting of 71 open source projects hosted at the SourceForge web site. The 71 projects in the sample were chosen (in January 2000) from more than 31,000 projects that were listed at the SourceForge web site at

the time by selecting the most active projects in the top-level list of "topics" at SourceForge.[13] This sample was observed over an eighteen-month period from January 2002 through the middle of 2003, with data collected at two-month intervals.[14] Hence, there are nine observations on each project. Although we only have data on a relatively small sample of the projects hosted SourceForge, the sample is unique because of data on lines of code as well as data on different versions of the program. The latter is a potentially important control variable, since a change in version may necessitate additional lines of code.

Our data set contains information on the size of the open source projects in the form of source lines of code (SLOC). Using SLOC as a performance measure is not always ideal; nevertheless, this performance measure is employed in the profession and the literature.[15] For our purposes, SLOC is in fact an ideal measure, because we want to measure the effort that is put into the project, rather than whether a project succeeds.

Variables Employed in the Analysis

Data were collected on the SourceForge hosted open source projects over an eighteen-month period at two-month intervals. Hence there are nine observations on each project. The following variables are available for the study.

- SLOC - the source lines of code at each point in time for each project.
- CONTRIBUTORS – the number of contributors to the project at each point in time for each project, i.e., the number of names on the list of contributors.[16]
- OUTPUT PER CONTRIBUTOR – the ratio of SLOC/CONTRIBUTORS.
- We have several variables that identify how restrictive is the license:[17]

[13] SourceForge lists activity levels for projects on its website. Activity level is an internal measure based on page views, downloads, Bugs, and Support Patches, etc. Hence, it is a measure that includes both inputs and outputs.

[14] We are grateful to NERA for providing us with the data. Although the projects were selected a couple of years before the data collection began, this does not cause a selection bias.

[15] According to "Cost Estimation and Project Management," which is available at available at http:///www.comp.glam.ac.uk/pages/staff/bfjones/ils/cocomo.htm, the most common measure of productivity is lines of code. Leppämäki and Mustonen (2003) use this measure as a proxy for output in their theoretical paper.

[16] All open source projects in our sample list their contributors' email addresses, and the algorithm employed for counting contributors was designed to control for contributors who list multiple email addresses.

1. RESTRICTIVEV – This variable takes on the value one if the project is licensed under the General Public License (GPL). Otherwise, this variable takes on the value zero.
2. RESTRICTIVEM –This variable takes on the value one if the project uses any one of the following licenses: LGPL, Mozilla, MPL, or NPL. Otherwise, the variable takes on the value 0.
3. RESTRICTIVE – This variable takes on the value one if the license is either very restrictive (RESTRICTIVEV=1) or moderately restrictive (RESTRICTIVEM=1).
4. NONRESTRICTIVE - This variable takes on the value one if the license employed is a BSD-type license.

- NEW VERSION – This variable takes on the value one if a new version of the program was developed in the two month period between observations.
- ADVANCED STAGE – The variable takes on the value one if the project was either in stage 5 or stage 6, where stage 5 is defined to be "Production/Stable" and stage 6 is defined to be "Mature," and zero otherwise.[18]
- Operating systems:
    1. LINUX - This variable takes on the value one if the project is written for the Linux operating system and takes on the value zero otherwise.
    2. MICROSOFT – This variable takes on the value one if the project is written for one of the Microsoft operating systems and takes on the value zero otherwise.
    3. POSIX – This variable takes on the value one if the project is written for the POSIX standards and takes on the value zero otherwise.
    4. MAC - This variable takes on the value one if the project is written for the MAC operating system and takes on the value zero otherwise.

[17] The empirical results suggest that there is little difference between very restrictive and moderately restrictive licenses. Hence we generally use the variable RESTRICTIVE. Nevertheless, we also discuss how the results change when we delineate restrictive licenses into very restrictive and moderately restrictive licenses.

[18] Each project was rated by stage of production, ranging from 1 to 6. Nearly all of the projects in our sample were in stages 3-6. We employ the variable in this manner, because some projects listed multiple stages at a single point in time, while other projects listed a single stage. Only a few projects that listed a single stage at each point in time changed stages during our time period.

5. BSD OS - This variable takes on the value one if the project is written for one of the BSD Unix-like operating systems (i.e., FreeBSD, NetBSD, or OpenBSD) and takes on the value zero otherwise.
6. SUN OS - This variable takes on the value one if the project is written for the SUN operating system (Solaris, a variant of Unix) and takes on the value zero otherwise.
7. OSI - This variable takes on the value one if the project is independent of any operating system and takes on the value zero otherwise.
8. LINUXONLY – This variable takes on the value one if the project was written solely for the Linux operating system and takes on the value zero otherwise. MICROSOFTONLY and POSIXONLY are defined similarly.

- SINGLE OS – This variable takes on the value one if the project was written for a single operating system and zero otherwise.
- AGE – This variable is the age of the project in years.
- DESKTOP – This variable takes on the value one if the intended audience is end users, i.e., users with desktops or laptops, and zero if the intended audience does not include end users.
- SYSTEM – This variable takes on the value one if the intended audience is system administrators and zero if the intended audience does not include system administrators.
- DEVELOPER - This variable takes on the value one if the intended audience is developers and zero if the intended audience does not include developers.
- LANGUAGE – This variable takes on the value one if the program is written in C or C++ and 0 otherwise.

Control Variables

It is difficult to compare lines of code between “high” and “low” level programming languages. This is because “lower level” programming languages have more lines of code and take longer to develop than higher level programming languages. Fortunately, every product in our sample uses high level programming languages. Despite the homogeneity in the level of the programming languages, there are likely differences in lines of code across different programming languages. Hence

we will also examine the large subset of the data where the programs were written in either C or C++. Of the 71 SourceForge projects in our sample, 56 are written in either C or C++.[19]

In the empirical work, we also control for the operating system that the project employs, as well as whether the project was written for single or multiple operating systems. We control for the age of the project and program type by including information about whether the program is intended for developers, system administrators, or end users. We also control for whether the observation represents a new version of the program, since a new version may necessitate additional lines of code.

Descriptive statistics

Descriptive statistics are shown in Table 1. Table 1 shows that 62% of all projects in the sample employed the GPL license. The table shows that the categories RESTRICTIVE and NONRESTRICTIVE are nearly exclusive. Indeed, only two projects in the sample offered both restrictive and nonrestrictive licenses.[20]

Table 1 also shows that the most popular "operating systems" are the LINUX, MICROSOFT, and POSIX operating systems.[21] Indeed one of these operating systems is employed in every project in our sample except for those projects that are exclusively operating system independent (OSI). Hence we focus on these operating systems in the analysis. Table 2 shows the breakdown of the projects according to these operating systems.

Table 3a shows the differences on key variables between projects with restrictive licenses and projects with non restrictive licenses. The table shows that the mean output (source lines of code) and the mean number of contributors are much greater for projects that use non-restrictive licenses.[22]

[19] Some of the programs in our sample are available in more than one (high level) programming language.
[20] The two projects were available under both licenses throughout the period for which we have data.
[21] It is our understanding that POSIX isn't technically an operating system, but rather a set of standards. Nevertheless, we'll refer to POSIX as an operating system. No projects were written for both LINUX and POSIX operating systems.
[22] Lerner and Tirole (2005) find that the activity level is higher for projects "that are not highly restrictive and (on a less consistently statistically significant basis) not restrictive." Since "activity" is a mixture of both input and output, it is hard to compare this with our main result. Lerner and Tirole (2005) do not have data on lines of code.

The most striking difference in Table 3a is that the output per contributor is approximately 6642 for the programs employing non-restrictive licenses and approximately 2319 for the programs employing restrictive licenses. In our formal empirical work, we will show that this raw correlation in the data remains even after we control for all other factors that might affect output per contributor. Hence although we choose to estimate a particular specification, the main result that the mean effort (output) per contributor is significantly greater for projects that use non-restrictive licenses is completely robust to alternative specifications; that is, raw correlations in the data drive the results.

Table 3b provides summary data for projects written for developers. The summary data in Table 3b are consistent with the Lerner and Tirole (2005) result that programs written for developers tend to use less restrictive licenses and employ commercial operating systems. The table shows that 80 percent of the programs not written for developers employ very restrictive licenses, while only 52 percent of the programs written for developers employ very restrictive licenses.

Additionally 43 percent of programs written for developers are written for a Microsoft operating system, while 26 percent of programs not written for developers employ a Microsoft operating system. Similarly 44 percent of programs written for developers employ a Linux operating system, while 64 percent of programs not written for developers are written for the Linux operating system.

4. Empirical Results

The initiators of an open source project likely first choose the license to employ, the operating system, and the intended audience, as well as other characteristics of the project. Once these attributes have been chosen, they typically do not change.[23] Indeed, these variables do not change at all over time in our data set. Hence, although these variables (characteristics) are clearly not exogenous, they are chosen at an earlier (initial) stage.

Once the characteristics of the project have been chosen, contributors join, code is written, more contributors join, more code is written, etc. Hence it is likely that both the number of contributors and the size of the program are endogenously

[23] There are a few well-known cases in which the license type was changed, but no changes in license type were made in the projects in our sample.

determined. Our dependent variable in the model below is the ratio of these two endogenous variables. Thus, the dependent variable (output per contributor) is clearly endogenous, while it's reasonable to assume that the other variables are predetermined, that is chosen at an earlier stage.

In the model below, we examine factors that explain output per contributor. We have a panel data set, that is, nine observations for nearly all projects over an eighteen month time period. Hence we employ a random effects model of the form

$$y_{it}=X_i \beta + \mu_i + \varepsilon_{it}, \quad (1)$$

where the index "i" represents the project and index "t" represents the time period. y_{it} is the output per contributor in project i at time t, X is a matrix that contains the characteristics of each project (license type, operating system, etc.), β measures the effect of these characteristics, μ_i is the random effect of project i and ε_{it} is a white noise error term.

Alternatively, we could have employed a fixed effects model, but the random effects model is appropriate since the 71 projects are a random sample from the larger population of projects hosted at SourceForge.[24] In any case, the main results are robust to employing a fixed effects model. The main results are also robust to looking at cross-sectional data as well, i.e., data at a single point in time.

4.1 The Effect of Restrictive Licenses

Table 4 presents three regressions using the full data set. The main result in Table 4 is as follows:

Observation 1: After controlling for operating system, audience, etc., projects that employ restrictive licenses result in significantly less effort (output) per contributor than projects that do employ less restrictive licenses.

[24] The Breusch and Pagan (BP) Lagrange multiplier test and (ii) the Hausman (HM) specification test indicate that the random effects model is appropriate. The results are robust to using maximum likelihood estimation or generalized least squares (GLS) estimation. In order to be concise, we report only the GLS results.

As noted above projects with restrictive licenses are likely to have a more limited commercial potential. Our empirical result thus suggests a professional status, signaling, or intrinsic motivation (personal satisfaction) in the programmers' decisions to participate in a project with a restrictive license. Since the quality of one's contribution is not easily observable, professional status or signaling may be obtained simply by being on the list of contributors. In other words, "making the list" is the main reward or significance of the contribution. The analogy to academia is that being on the list is like being a co-author on a high-quality academic paper.

Note that unlike many of the other characteristics; the type of license does not "technically" affect the writing of code. This statistically significant result is likely "economically" significant as well: other things being equal, output per contributor is approximately 5000 lines less when a restrictive license is employed.

The result is very robust to different specifications as shown in Table 4. The difference between regressions 1 and 2 in Table 4 is that in table 2, RESTRICTIVEV and RESTRICTIVEM are employed rather than restrictive. Similarly, regression 3 in Table 4 shows that result is robust to employing LINUXONLY, MICROSOFTONLY, and POSIXONLY, rather than LINUX, MICROSOFT and POSIX.[25] The result is also robust to adding LANGUAGE to the set of explanatory variables as well.

In the first regression in Table 5, we present regression results for projects written for either C or C++. Table 5 shows that the estimated coefficient on RESTRICTIVE is of the same order of magnitude as the estimated coefficients on RESTRICTIVE in Table 4 (-6,207, t=-2.81).

An interesting question is whether this result is due to the fact that there are more contributors or less code per project, or some combination of both. This question can be examined by considering medians rather means. The median output per contributor is approximately 2125 for the programs employing non-restrictive licenses and approximately 1367 for the programs employing restrictive licenses. Hence, even in the case of medians, there is still a relatively large difference in output per contributor. This suggests that our main result is quite robust.

Further examination shows that there is a relatively small difference in the median lines of code per project: the median is approximately 52,978 source lines of

[25] Of course in this case, SINGLE OS has to be excluded due to multicollinearity.

code for programs employing restrictive licenses and 60,309 source lines of code for projects that do not employ such licenses. On the other hand, there is a very large difference in the median number of contributors: 35 for projects that employ restrictive licenses and 13 for projects that do not employ restrictive licenses. This provides support for the ideological or status/signaling incentive to be a contributor to a project with a restrictive license.[26]

4.2 The Effect of Program Type

We included the variables SYSTEM, DEVELOPER, and DESKTOP in the regressions in Table 4 in order to control for the heterogeneity among projects.

Observation 2: Output per contributor in projects oriented towards end users (DESKTOP) is significantly lower than that in projects for developers.

There might be several possible explanations of this result. It is possible that in some of the projects for end users, developers may modify the software for private use. It is also possible that projects for end users are less commercial than projects for developers.

In order to explore this issue further, the second regression in Table 5 includes only projects written for developers. Since many of the projects are for multiple audiences, we still control for end users and system operators. The difference in output per contributor between non-restrictive and restrictive licenses is approximately 8000 lines of code.

Observation 1 suggests that projects that employ restrictive licenses have much smaller output per contributor than projects that do not employ restrictive licenses. Since programs written for developers tend to be commercial, these results reinforce the notion that contributors in non-commercial programs do so for a signaling, status or ideological motive.

4.3 The Effect of Operating Systems

We now turn to discuss the role of the type of operating system in determining output per contributor. The relevant regressions are given in Table 4.

[26] Another possibility is that projects that have chosen a restrictive license are perhaps more "community-oriented" and thus more generous in giving recognition than commercial projects.

Observation 3: Projects written for the Linux operating system have lower output per contributor than projects written for other operating systems.

Observation 3 may indicate on the same ideological or intrinsic motivation in the programmer's decision to participate in an open source project. A project that is written for the Linux operating system may indeed involve a greater ideological motivation. But on the other hand we cannot exclude the possibility that these differences are due to technical issues, that is, programs written for the LINUX operating system require less code overall.

The effect is statistically significant in all of the regressions in Table 4 and the difference is approximately 2000 lines per contributor. This difference is smaller than the difference in output per contributor for different license types. In the case of programs written in C/C++ (first regression in Table 5), the magnitude of the effect is similar to the regressions in Table 4.

In the case of projects written for developers (second regression in Table 5), the LINUX effect (-1,556, -1.04) is statistically insignificant, even though the estimated magnitude is fairly large. In this regression, the estimated coefficient on Microsoft is positive and statistically significant (4,024, t=1.82).[27]

4.4 Other factors that affect the size of open source programs

We now discuss the other factors that affect the output per contributor of open source programs. The coefficient on NEWVERSION is negative as expected, although not statistically significant in either Table 4 or Table 5. Hence there is some (weak) evidence that per contributor output is slightly lower around the time when a new version is introduced. The other factors generally do not explain the variance in output per contributor, but we include them as controls. There is a potential sample selection issue associated with AGE, since age is the initial date at which the project was first hosted at SourceForge. If the project began elsewhere, it is indeed older than the age we employ. This likely does not have any effect on our main results because actual age is likely quite highly correlated with the age from when the project first appeared at SourceForge. Additionally, the estimated coefficient on AGE is virtually

[27] The regressions in Tables 4 and 5 show that the POSIX effect is similar in magnitude to the LINUX effect.

zero, suggesting that age is not a factor in explaining the variance in output per contributor.

5. Further Discussion

Our main result – that output per contributor in open source projects is much higher when licenses are less restrictive and more commercially oriented – suggests a status, signaling, or intrinsic motivation for participation in open source projects with restrictive licenses. We believe that this result helps shed some light on the development of a key part of the open source software industry, namely on the large group of open source projects that are not well-known.

While SourceForge is the largest open source development site and hosts a very large number of open source projects, the well-known open source projects such as Linux (operating system), Apache (web server), Bind (A domain name software), Perl (a scripting language), and Sendmail (an E-mail server) are not hosted at SourceForge. It is possible that the well-known open source projects differ from the projects hosted at SourceForge in important ways. For example, of the above open source programs, only LINUX was exclusively released under the GPL. The other four projects use BSD-type licenses.[28] It would be interesting to empirically compare well-known (successful) open source projects (which are not typically hosted at SourceForge) with projects hosted at SourceForge.

Data on "who contributed what" over time are available at SourceForge on a project by project basis. Hence, a potentially interesting future project would be to examine whether contributors to open source projects are indeed made close to a threshold level and whether contributions decrease (or even stagnate) once the level has been reached.

[28] It is our understanding that Perl uses both a BSD-type license and a GPL license.

References:

Bonaccorsi, A., and C. Rossi (2002), "Licensing Schemes in the Production and Distribution of Open Source Software: An Empirical Investigation" available at http://opensource.mit.edu/papers/bnaccorsirossilicense.pdf.

Ghosh, R., Glott, R., Krieger, B., and G. Robles (2002), "Free/Libre and Open Source Software: Survey and Study, Part IV: Survey of Developers," FLOSS Report, International Institute of Infonomics, University of Maastricht, The Netherlands, available at http://www.infonomics.nl/FLOSS/report/Final4.htm.

Hann, I., Roberts, J., and S. Slaughter (2002) "Delayed Returns to Open Source Participation: An Empirical Analysis of the Apache HTTP Server Project, Carnegie Mellon University mimeo.

Halloran, T., and W. Scherlis (2002), "High Quality and Open Source Software Practices, mimeo, available at http://opensource.ucc.ie/icse2002/HalloranScherlis.pdf.

Harhoff, D., J. Henkel, and E. von Hippel (2003), "Profiting from voluntary spillovers: How users benefit by freely revealing their innovations, *Research Policy* 32: 1753-1769.

Hars, A., and S. Ou (2001), "Working for free? - Motivations for participating in open source projects," *International Journal of Electronic Commerce*, 6: 25-39

Hertel, G., Niedner, S. and S. Herrmann (2003), "Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel," *Research Policy*, 32, 1159-1177.

Johnson, J. (2002), "Open Source Software: Private Provision of a Public Good," *Journal of Economics & Management Strategy*, 11: 637-662.

Johnson, J. (2005), "Collaboration, Peer Review and Open Source Software" mimeo, Cornell University.

Kuan, J. (2002), "Open Source Software as Lead User's Make or Buy Decision: A Study of Open and Closed Source Quality," Stanford Institute for Economic Policy Research mimeo.

Lakhani, K., and R. Wolf (2005), "Why Hackers Do What They Do: Understanding Motivation and Efforst in Free Open Source Projects, In: Feller/Open, J. Fitzgerland, S. Hissam, K. Lakhani (eds.), Perspectives on Free and Open Source Software, MIT Press, Cambridge.

Leppämäki, M., and M. Mustonen (2003), "Spence Revisited - Signaling with Externality: The Case of Open Source Programming," University of Helsinki Discussion Paper 558.

Lerner, J., and J. Tirole (2002), "Some Simple Economics of Open Source" *Journal of Industrial Economics*, 52: 197-234.

Lerner, J., and J. Tirole (2005), "The Scope of Open Source Licensing," forthcoming, *Journal of Law, Economics and Organization*.

Raymond, E. (2000), "The Cathedral and the Bazaar, available at http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

Shah, S. (2002), "A Summary of 'Open Source Software as Lead User's Make or Buy Decision: A Study of Open and Closed Source Quality," TIIP Newsletter, available at http://www.researchoninnovation.org/tiip/archive/kuan.htm.

Stallman, R., 1999, "The GNU Operating system and the Free Software Movement," in Dibona, C., Ockman, S., and M. Stone editors, *Open Sources: Voices from the Open Source Movement,* O'Reilly, Sepastopol, California, available at http://www.oreilly.com/catalog/opensources/book/stallman.html.

Table 1: Descriptive Statistics (N=636)[29]

VARIABLE	MEAN	STD. DEV.	MIN	MAX
SLOC	120,355.10	266,341.30	275	2,337,136
CONTRIBUTORS	83.32	157.50	1	1091
OUTPUT PER CONTRIBUTOR	3358.91	6485.26	68.75	55024.50
AGE	2.06	0.73	0.027	3.50
NEW VERSION	0.50	0.50	0	1
ADVANCED STAGE	0.58	0.49	0	1
LINUX	0.50	0.49	0	1
MICROSOFT	0.37	0.48	0	1
OS INDEPENDENT	0.30	0.46	0	1
POSIX	0.26	0.44	0	1
BSD	0.13	0.33	0	1
MAC	0.08	0.28	0	1
SUN	0.08	0.27	0	1
LINUXONLY	0.13	0.34	0	1
POSIXONLY	0.063	0.24	0	1
MICROSOFTONLY	0.13	0.33	0	1
SINGLE OS	0.30	0.46	0	1
RESTRICTIVEV	0.62	0.49	0	1
RESTRICTIVEM	0.18	0.39	0	1
RESTRICTIVE	0.76	0.42	0	1
NONRESTRICTIVE	0.27	0.44	0	1
DESKTOP	0.64	0.48	0	1
SYSTEM	0.31	0.46	0	1
DEVELOPER	0.65	0.48	0	1
LANGUAGE	0.79	0.41	0	1

[29] There are potentially 639 observations. We lack some data on three of the observations.

Table 2: Number of projects per operating system

Operating System	Number of Projects
LINUX ONLY	9
POSIX ONLY	9
MICROSOFT ONLY	4
LINUX + MICROSOFT	16
POSIX + MICROSOFT	7
LINUX + OTHER OS	12
POSIX + OTHER OS	2
EXCLUSIVELY OSI	12

Table 3a: Descriptive statistics according to license use

Projects that employ restrictive licenses (N=483)				
VARIABLE	**MEAN**	**STD. DEV.**	**MIN**	**MAX**
SLOC	81,531.07	99,870.05	3619	530,314
CONTRIBUTORS	64.20	78.38	1	469
OUTPUT PER CONTRIBUTOR	2,318.63	2,462.53	250.95	10,904.8
DESKTOP	0.61	0.49	0	1
SYSTEM	0.24	0.45	0	1
DEVELOPERS	0.63	0.48	0	1
LINUX	0.52	0.50	0	1
MICROSOFT	0.42	0.49	0	1
POSIX	0.28	0.49	0	1
OSI	0.27	0.44	0	1
SINGLE OS	0.35	0.48	0	1
LANGUAGE	0.85	0.85	0	1
Projects that do not employ restrictive licenses (N=153)				
SLOC	242,917.40	494,786.20	275	2,337,136
CONTRIBUTORS	143.70	281.64	1	1091
OUTPUT PER CONTRIBUTOR	6,641.61	11,924.35	68.75	55,024.50
DESKTOP	0.70	0.46	0	1
SYSTEM	0.28	0.49	0	1
DEVELOPERS	0.71	0.46	0	1
LINUX	0.46	0.50	0	1
MICROSOFT	0.24	0.43	0	1
POSIX	0.18	0.38	0	1
OSI	0.48	0.50	0	1
SINGLE OS	0.24	0.43	0	1
LANGUAGE	0.59	0.49	0	1

Table 3b: Descriptive statistics according to audience

Projects for developers (N=414)				
VARIABLE	**MEAN**	**STD. DEV.**	**MIN**	**MAX**
SLOC	165584.00	320592.70	4044	2337136
CONTRIBUTORS	102.29	189.79	1	1091
OUTPUT PER CONTRIBUTOR	4238.83	7730.89	250.95	55024.5
DESKTOP	0.52	0.50	0	1
SYSTEM	0.27	0.44	0	1
LINUX	0.44	0.50	0	1
MICROSOFT	0.43	0.50	0	1
POSIX	0.30	0.46	0	1
OSI	0.44	0.50	0	1
RESTRICTIVE	0.74	0.44	0	1
RESTRICTIVEV	0.52	0.50	0	1
RESTRICTIVEM	0.28	0.45	0	1
SINGLE OS	0.24	0.43	0	1
LANGUAGE	0.76	0.43	0	1
Projects not for developers (N=222)				
SLOC	36009.47	27832.08	275	105345
CONTRIBUTORS	47.95	45.10	1	237
OUTPUT PER CONTRIBUTOR	1717.07	2235.70	68.75	10699.5
DESKTOP	0.85	0.36	0	1
SYSTEM	0.39	0.49	0	1
LINUX	0.64	0.48	0	1
MICROSOFT	0.26	0.44	0	1
POSIX	0.17	0.38	0	1
OSI	0.081	0.273577	0	1
RESTRICTIVE	0.80	0.402921	0	1
RESTRICTIVEV	0.80	0.402921	0	1
RESTRICTIVEM	0	0	0	0
SINGLE OS	0.48	0.50	0	1
LANGUAGE	0.84	0.37	0	1

Table 4: Dependent Variable: OUTPUT PER CONTRIBUTOR (Full Data Set)

	Regression 1		Regression 2		Regression 3	
Independent Variables	Coeff.	T stat	Coeff.	T stat	Coeff.	T stat
CONSTANT	12640.86	5.94	13157.77	6.11	10864.14	5.15
RESTRICTIVE	-4840.83	-2.90			-4926.39	-2.93
RESTRICTIVEV			-3811.57	-2.40		
RESTRICTIVEM			-6135.64	-2.79		
DESKTOP	-2943.94	-1.97	-4141.83	-2.48	-3062.4	-2.01
SYSTEM	-2433.84	-1.57	-2465.52	-1.59	-2523.48	-1.60
DEVELOPERS	168.912	0.27	258.93	0.41	134.7423	0.21
LINUX	-2009.03	-3.39	-2070.84	-3.49		
LINUXONLY					-2040.09	-3.25
MICROSOFT	323.17	0.54	265.13	0.44		
MICROSOFTONLY					813.44	1.28
POSIX	-2256.97	-2.43	-2233.54	-2.40		
POSIXONLY					-2383.97	-1.08
OSI	-2112.14	-3.42	-2067.32	-3.34	-1018.17	-1.52
AGE	-35.82	-0.37	-35.64	-0.37	-32.62	-0.34
NEWVERSION	-163.88	-1.67	-161.88	-1.65	-151.72	-1.55
ADVANCED STAGE	-282.25	-0.57	-219.23	-0.44	-222.92	-0.49
SINGLE OS	-1901.03	-3.28	-1925.04	-3.32		
Number of observations	636		636		636	
	χ^2 stat	p val	χ^2 stat	p val	χ^2 stat	p val
BP Lagrange Multiplier Test[30]	2321.65	0.00	2328.33	0.00	2318.73	0.00
HM Test[31]	9.08	0.43	8.12	0.52	6.03	0.54

[30] Formally, the null hypothesis for the Lagrange Multiplier test is that var(u)=0. If the null hypothesis is rejected, the random effects model is appropriate. The chi-squared statistics in tables 4 and 5 clearly indicate that the null hypothesis should be rejected in all of the regressions.

[31] Formally the HM statistic tests the null hypothesis that the independent variables (X_i) are not correlated with μ_i , the random effect of project i. The chi-squared statistics in tables 4 and 5 clearly indicate that the null hypothesis should not be rejected in any of the regressions. Hence the evidence indicates that X_i and μ_i are uncorrelated.

Table 5: Dependent Variable: OUTPUT PER CONTRIBUTOR

	Projects Written in C/C++		**Projects for Developers**	
Independent Variables	Coefficient	T-statistic	Coefficient	T-statistic
CONSTANT	13326.36	5.14	14698.89	4.01
RESTRICTIVE	-6206.95	-2.81	-7905.71	-3.37
DESKTOP	-2263.2	-1.27	-2899.94	-1.44
SYSTEM	-1296.64	-0.68	-1395.61	-0.6
DEVELOPERS	61.88	0.09		
LINUX	-2144.37	-3.27	-1555.53	-1.04
MICROSOFT	356.69	0.54	4023.50	1.82
POSIX	-2444.49	-2.34	-3804.84	-1.58
INDEPENDENT OS	-2060.47	-3.06	-1561.27	-0.95
AGE	36.611	0.31	34.51	0.26
NEWVERSION	-186.27	-1.58	-117.94	-0.88
ADVANCED STAGE	-409.64	-0.75	-2668.11	-1.21
SINGLE OS	-1915.88	-3.03	-1045.17	-0.36
Number of observations	501		414	
	χ^2 stat	p value	χ^2 stat	p value
BP Lagrange Multiplier Test	1806.54	0.00	1483.04	0.00
HM Test	9.27	0.41	4.06	0.40