

The Dynamics of Open Source Contributors

Josh Lerner, Parag Pathak, and Jean Tirole^{*}

December 9, 2005

^{*}Harvard University and NBER; Harvard University; and University of Toulouse and Massachusetts Institute of Technology. We thank for research assistance Vakha Elmurzev, Lee Gao, James Hunter, Payal Loungani, Susanna Kim, Qicheng Ma, and John Sheridan. Helpful comments were received from participants in the Toulouse Network on Information Technology's Fall 2005 meeting and a number of practitioners. Jeff Bates was extremely helpful in regard to SourceForge access. We thank the Harvard Business School, the National Science Foundation, and the Toulouse Network on Information Technology for financial support.

1. Introduction

This paper seeks to understand the dynamics of contributions to open source projects. This is an interesting question because of the substantial differences between these projects and traditional innovative efforts in private firms, which usually pay their workers, direct and manage their efforts, and control the output and intellectual property thus created.

By way of contrast, contributions to open source projects are made by a diverse array of individual contributors and for-profit corporations, who in many cases must agree to make all enhancements to the original material widely available for nominal cost. The dynamics of open source projects, and the key insights from earlier research, are discussed in Section 2.

To frame these questions, we sketch our theoretical predictions in Section 3. In a setting where there are two types of contributors, corporations and individual “hobbyists,” we posit that there are three forms of benefits from contributing to open source projects: commercial, signaling (either to peers or prospective employers), and exogenous (e.g., ideology and meeting specific programming needs). We predict that the share of corporate contributions should be more sensitive to the growth of the project than those of the hobbyists. This effect should be particularly strong when the commercial prospects of the project are favorable: for instance, when venture capitalists are financing companies committed to the open source project or when the code base is substantial already.

We then seek to test these ideas empirically. To this end, we construct a panel data-set of activity in one hundred open source projects between 2001 and 2004. As

described in Section 4, we compile the contributors to each version of each of these projects, and classify them into several broad classes.

In the analysis in Section 5, we find several patterns consistent with theoretical predictions:

- The volume of contributions by corporate contributors is greater for projects where success—and in particular, commercial success—seems more likely. This includes later versions of open source projects, as well as (less consistently) larger and faster-growing ones, as well as those where venture capitalists have funded related firms.
- The sensitivity of contributions by corporate affiliates to project growth is several times greater than that of hobbyists, a difference that is consistently statistically significant.
- The difference between the sensitivity of contributions of hobbyists and corporate affiliates to project growth is greatest in projects with the greatest probability of commercial applications, including larger projects and those with unrestrictive licenses.

This paper builds on theoretical works that have sought to compare the relative merits of open source and proprietary software. Among others, Casadesus-Masanell and Ghemawat (2003), Gaudoul (2004), and Johnson (2002) present models that predict when open source products will be superior to traditional commercially developed software and *vice versa*. But the question of the mixture of contributors to open source projects has not examined here.

There is also a limited amount of empirical economics literature on open source contributions, most of which has focused on understanding the motivations of individual contributors. The sole non-survey study we are aware of, by Hann, et al. (2004), examines contributors to the Apache project, drawing on a wide variety of project records and survey data for 147 contributors to the project. The authors then estimate a series of regressions, in which they use the logarithm of earnings in a given year as the dependent variable, and information on the respondents' background, work experience, contributions and current position to the Apache project in the previous year, and individual fixed effects as independent variables. The results suggest that the sheer volume of contributions to the Apache project have little impact on salary. But individuals who attain high rank in the Apache organization enjoy wages that are 14 to 29 percent higher, whether or not their work directly involves the Apache program. Academics have also attempted to understand motivations of those who work on open source projects through surveys. Given the inherent subjectivity of these assessments and the self-serving biases in reporting, the low response rates that many of these surveys have obtained, and the sensitivity of some of these questions, it is perhaps not surprising that self-reported motivations vary considerably across studies. For instance, Haruvy, Wu and Chakravarty (2003) find that commercial objectives—particularly, the promise of higher future earnings—are an important driver of contributions to open source projects, while Lakhani and von Hippel (2003) suggest that the overwhelming driver of open source contributors is the need to solve their own specific programming needs.

2. Incentives and Open Source Projects¹

The decision to contribute without pay to freely available software may seem mysterious to economists. However, the standard framework of labor economics can be adapted to capture activity in the open source environment (Lerner and Tirole, 2002).

The unpaid programmer working on an open source software development project faces a variety of benefits and costs. The programmer incurs an opportunity cost of time, which can manifest itself in different ways. For example, a programmer who works as an independent on open source projects forgoes the monetary compensation that could otherwise be earned by working for a commercial firm or a university. For a programmer with a commercial company, university or research lab affiliation, the opportunity cost of working on open source software comes from not focusing on other tasks. For example, the academic's research output may sag and the student's progress towards a degree slow down.

Several short-or long-run benefits may counter these costs. First, open source programmers may improve rather than reduce their performance in paid work. This outcome is particularly relevant for system administrators looking for specific solutions for their company. Second, the programmer may find intrinsic pleasure if choosing a “cool” open source is more fun than a routine task set by an employer. Third, in the long run, open source contributions may lead to future job offers, shares in commercial open source-based companies, or future access to the venture capital market, and last (but not

¹This section is based on Lerner and Tirole (2005a).

least) ego gratification from peer recognition. Of course, different programmers may put different values on monetary or personal payoffs, and on short-term or long-term payoffs.

Economic theory suggests that long-term incentives are stronger under three conditions: 1) the more visible the performance to the relevant audience (peers, labor market, and venture capital community); 2) the higher the impact of effort on performance; 3) the more informative the performance about talent (for example, Holmström, 1999). The first condition gives rise to what economists call “strategic complementarities.” To have an “audience,” programmers will want to work on software projects that will attract a large number of other programmers. This argument suggests the possibility of multiple equilibria. The same project may attract few programmers because programmers expect that other programmers will not be interested; or it may flourish as programmers (rationally) have faith in the project.

To compare programmers' incentives in the open source and proprietary settings, we need to examine how the features of the two environments shape incentives. From the standpoint of the individual, commercial projects typically offer better current compensation than open source projects, because employers are willing to offer salaries to software programmers in the expectation that they will capture a return from a proprietary project. Yet, even commercial firms that compensate programmers may want their employees to work on open source projects. For instance, the activity will allow firms to fix bugs and customize the product to one's own ends for the programmer. Second, open source code may already be familiar to programmers: because it is freely available to all, it can be used in schools and universities for learning purposes, thus

creating an “alumni effect.” Finally, the initial cost of accessing the software is very low, an important consideration for new firms.

When we consider the delayed rewards of working on an open source project, the ability to signal a high level of competence may be stronger in the open source mode for three reasons. First, in an open source project, outsiders can see the contribution of each individual, whether that component “worked,” whether the task was hard, if the problem was addressed in a clever way, whether the code can be useful for other programming tasks in the future, and so forth. Second, the open source programmer takes full responsibility for the success of a subproject, with little interference from a superior, which generates information about ability to follow through with a task. Finally, since many elements of the source code are shared across open source projects, more of the knowledge they have accumulated can be transferred to new environments, which makes programmers more valuable to future employers.

Commercial companies may interact with an open source project in a number of ways. While improvements in the open source software are not appropriable, commercial companies can benefit if they also offer expertise in some proprietary segment of the market which is complementary to the open source program. Firms may temporarily encourage their programmers to participate in open source projects to learn about the strengths and weaknesses of this development approach. For-profit firms may compete directly with open source providers in the same market. Finally, commercial companies may interface with the open source world because it generates good public relations with programmers and customers.

A for-profit firm that seeks to provide services and products which are complementary to the open source product, but are not supplied efficiently by the open source community, can be referred to as “living symbiotically.” IBM, which has made open source software into a major focus for its consulting business, exemplifies this approach. A commercial company in this situation will want to have extensive knowledge about the open source movement, and may even want to encourage and subsidize open source contributions, both of which may cause it to allocate some programmers to the open source project. Because firms do not capture all the benefits of the investments in the open source project, however, the free-rider problem often discussed in the economics of innovation should apply here as well. Subsidies by commercial companies for open source projects should remain somewhat limited.

The code release strategy arises when companies release some existing proprietary code and then create a governance structure for the resulting open source development process. This strategy is to giving away the razor (the released code) to sell more razor blades (for instance, the related consulting services that IBM and HP hope to provide). In general, it will make sense for a commercial company to release proprietary code under an open source license if the increase in profit in the proprietary complementary segment offsets any profit that would have been made in the primary segment, had it not been converted to open source. Thus, the temptation to go open source is particularly strong when the product is lagging behind the leader and making few profits, but the firm sees a possibility that if the released code becomes the center of an open source project and is utilized more widely, the profitability of the complementary segment will increase.

3. Theoretical Predictions

A stylized setting allowing a comparison of the benefits and costs of contributing to a project for different groups of contributors involves two periods and two types of contributors: corporations and “hobbyists.” We assume that there are three types of benefits:

1. Commercial benefits accrue at the end of date 2 and increase with the total size of code at that date. The mean level of commercial benefits may depend on the level of venture capital support, the end-user orientation, the (lack of) restrictiveness of the license, or the size of the project.
2. Signaling benefits (ego gratification, career concerns) are forward looking and depend on the current and future visibility of the project. A key element of the model is that benefits at date 1 are proportional to the number of new contributions both in the concurrent period and in the future period (if any), so that there is a form of dynamic complementarity.
3. Exogenous benefits are benefits that are not or weakly related to the number of contributions. For instance, contributors may be motivated simply by ideology or by “scratching an itch.”

One simplifying assumption is that corporate contributors do not care about signaling benefits, while hobbyists do not anticipate commercial benefits. This allows us to keep the model tractable while generating some empirical predictions. More generally, we could allow contributors to put weight on the two benefits as long as corporate

contributors value primarily commercial benefits and hobbyists signaling benefits (among these two forms of benefits).

The costs of contributing are allowed to depend on the type of contributor. Since exogenous benefits are unrelated to the number of contributions, they will produce a level effect which will be netted out when the costs of contributions are considered. From the cost-benefit calculation, we can determine the level of contributors at the second date.

Under certain assumptions on the distribution of costs and on functional forms, the model implies a relationship between the relative sensitivity of contributions to the growth in contributions and the expected commercial benefits which depends on the overall level of exogenous benefits, the intensity of signaling benefits, and the expected commercial potential. In particular, when the expected commercial benefits increase, the relative sensitivity of contributions to current growth in contributions will also increase.

This relationship forms the basis of our empirical tests. We analyze how the difference in relative contribution sensitivity is related to measures of commercial viability. These measures include the level of venture capital support, the end-user orientation, the restrictiveness of the project license, and project size.

4. The Sample

In order to explore the dynamics of open source contributions, we built a panel data set of the contributors to approximately 100 open source projects (see Appendix). These projects are stratified to over-represent the largest open source projects. We extract the contributors to the project in each new official version of the program has been released, using a variety of text editing tools.

A. Selecting the Projects

The first question we faced related to the choice of projects that would constitute the sample. We wished to identify a sample of projects that would capture a representative cross-section of the open source universe. At the same time, we were aware that the distribution of activity in open source projects was highly skewed, with a small fraction of the projects representing the bulk of the activity.

To address these conflicting goals, we first identified a random sample of 78 projects based on the SourceForge database. (One was dropped from the SourceForge database during the sample period.) SourceForge is a free service that since 1999 has offered hosting and project administration tools to software development projects. SourceForge contained (as of May 2001, when the data were initially accessed) approximately thirty thousand projects. Essentially, it accepts listings of (and is willing to host) all open source projects that conform to the Open Source Definition discussed above.

Not all open source projects, however, are hosted on SourceForge. Many of the largest projects instead have their own web sites.² To this end, we identified twenty of the largest projects which operated independent web-sites.

Table 1 summarizes the projects, and highlights that they differed considerably in their size and other characteristics. Open source projects will typically periodically

²Other projects are hosted at smaller competing sites. These tend, however, to be much smaller: Savannah, often referred to as SourceForge's leading competitor, was one-fifth the size of SourceForge (as measure by number of active projects) in December 2005.

introduce new versions. The number of versions introduced between the beginning of data collection and July 2004 varied between one and twenty.³

B. Characterizing the Projects

We gathered a variety of data about each project. SourceForge undertakes a detailed classification of each project, including type of license employed and class of users targeted by the projects. While the non-SourceForge projects' web-sites did not always disclose this information, we did a series of Internet and news story searches to identify this information where possible. We identified whether venture capitalists had funded a company to commercialize or sell services related to each open source project using the Thomson Venture Economics and DowJones VentureSource databases, as well as press accounts.

We were particularly concerned with identifying projects that were profoundly influenced by a single corporation. As alluded to above, in some cases, a corporation will release some of its code as an open source project. In related cases, a single corporation may play a critical role in shaping a project as its major sponsor. We were concerned that the dynamics of these projects might be fundamentally different from the others. We identified spun-off projects and instances where a single corporation was a major sponsor of an open source project from news accounts and Internet searches.

C. Identifying Project Contributors

³We did an initial analysis using 20 SourceForge projects beginning in May 2001. In January 2002, we expanded the data collection to include the entire sample, which was tracked until July 2004.

The most challenging—and critical—measure was the number of contributions by each individual to the open source project. To undertake the analysis, we employed the “tarball” associated with each version of the project. This phrase refers to a file saved in the tar (Tape Archive) archive file format. These files are used widely to archive and unarchive files, that is, to accumulate a large collection of files into a single file, while preserving information such as user and group permissions, dates, and directory structures.⁴ From the tarball containing each version of the project, we extracted the number of distinct references to each individual.

Open source projects are scrupulous about keeping track of contributors, which reflects the fact that giving credit to authors is essential in the open source movement. This principle is included as part of the nine key requirements in the “Open Source Definition.”⁵ This point is also emphasized by Raymond (1999), who points out “surreptitiously filing someone’s name off of a project is, in cultural context, one of the ultimate crimes.” This point was also emphasized in our conversations with open source project managers and SourceForge officials.

Ideally, we would have measured the number of lines contributed by each party. This information, however, proved to be impossible to determine. Instead, we measured the number of additional mentions of individuals in each new version. From conversations with a number of open source project administrators, we learned that mention in a tarball is typically associated with a significant addition to the project, whether the contribution of code or (more rarely) the identification of an important bug.

⁴This definition is based on <http://en.wikipedia.org/wiki/Tarball> (accessed December 4, 2005).

⁵http://www.opensource.org/docs/definition_plain.php (accessed December 4, 2005).

Thus, it is important to acknowledge that when we use the phrase “contributions” below, we are using a proxy for our ideal measure.

It is important to also acknowledge that there is some degree of variation across projects in the manner in which they handle contributors. For instance, the standards for recognition of contributors differ across projects. In some cases, individuals who merely suggested “bugs” were recognized in the programs; in others, only those who made material code contributions were noted. The panel structure of the dataset is helpful in dealing with this complication.

The procedure we employed to extract the contributions to each version of the projects was as follows. For each project release, we downloaded the entire tarball archive. Then a script identified all instances where an e-mail address was found in the source code. To ensure that each e-mail was a valid one, we then matched the top-level domain to a list of valid top-level domains provided by the Internet Assigned Numbers Authority.⁶ Finally, to avoid double-counting e-mails, we manually checked the top 20 contributors for each project and identified instances where we could credibly identify an email address as referring to the same individual. For instance, in the project Cronos II, there were 42 instances of the e-mail petergozz@users.sourceforge.net, 422 instances of petergozz@users.sf.net, and 485 instances of petergozz@users.sourceforge.net. All of these observations were treated as the same individual making contributions. While we have not corrected all errors in the dataset, we believe we have corrected a majority of the obvious ones.

⁶The IANA maintains a list of valid Top Level Domains at <http://www.iana.org/cctld/cctld-whois.htm>

Panel A of Table 2 summarizes the contributors in the sample and shows that for the median project, the top 20 contributors contribute over 90% of all contributions. The table also shows the great diversity in the number of contributors, the contributions they make, and the concentration of the projects.

Table 2 shows that there can sometimes be negative growth in contributions as indicated by JFS having 343 fewer emails in a contribution. This complication is due to periodic “clean-ups” of open source projects. In approximately 7% of the new versions, the number of lines of code drops, often dramatically. In these cases, the total number of contributions in the tarball typically falls, as code and the associated credits are excised. In these instances, it is very difficult to identify new contributions. Thus, we deleted this subset of cases from the sample.

We sought to distinguish between individuals who were contributing code on their own behalf from those doing so as part of their employment. This is challenging because individuals may have multiple e-mail addresses: for instance, a contributor may make contributions by day as an IBM employee and at night from an individual account through his Internet Service Provider (ISP). While recognizing that no classification scheme will be perfect, we devised the following approach to identifying contributors in categories.

We divided the contributors into five classes based on their e-mail addresses. These are corporate employees, individual hobbyists, and three classes of otherwise other contributors: unidentified international contributors, and those from organizations with top-level domains (TLDs) denoted “.org” and “.net,” which frequently denote non-profit and technical web sites. We included as corporate contributors all those with a “.com”

address, excluding those sites used primarily as e-mail mailboxes, Internet Service Providers, or portals (e.g., “hotmail.com”). We also included overseas addresses that are associated with corporations (for instance, “co.uk” and “caldera.de”). We included as hobbyists contributions by individuals affiliated with universities and governments (again, employing both addresses with TLDs such as “.edu” and overseas domains like “umontreal.ca”), as well as those who made contributions from addresses associated with portals, ISPs, and mailboxes.⁷ The remaining categories—those from TLDs “.org” and “.net,” as well as the remaining international domains—were not classified in either category, but rather treated separately, because we were not able to readily assign them.

Panel B of Table 2 provides some more detail on this process. It highlights the mix of contributions by the five categories. The most contributions are by corporate affiliated, with those with TLD “.org” and unassigned international contributors the next largest categories. Table 3 provides a detailed breakdown of the TLDs in the sample.

5. The Analysis

In the empirical analysis, we proceed in three parts. First, we seek to understand the distribution of contributions to open source project by class of contributor, focusing—as in the theoretical discussion above—on contributions by corporations and “hobbyists.” Then, we examine the relative sensitivity of contributions to project growth by different

⁷One complication was posed by sites such as “aol.com,” which are used by both corporate employees and as an e-mail service. We treat these cases as corporate contributors. We have experimented with further portioning the corporate contributors into subcategories, where cases like “aol.com” will be considered to be separate. With this further breakout of the corporate sample, the qualitative results are similar.

classes of contributors. Finally, we examine how the sensitivity of these contributions varies with the features of the open source projects.

A. Analysis of Project Contributions

Table 4 presents our initial analysis, using the most direct measure: the number of contributions by each class of contributor for various classes of projects. This table presents the proportion of all contributions that are corporate. (Results looking at the ratio of contributions by corporate contributors and hobbyists generate similar results). The table also presents the result of F- and t-tests of the significance of the reported differences.

The analysis highlights that corporate contributions are more frequent for larger projects. The share of corporate contributions is twice as large in the largest quartile of projects than in the bottom. The pattern is similar, though somewhat less dramatic, when we compare the versions divided into quartiles based on their growth rates, defined here as the difference between the number of lines of code in the current and previous version. Both differences are highly statistically significant.

Patterns regarding license type and venture capital backing are less sharp. The share of corporate contributions is lowest among those projects with the most restrictive licenses (see Lerner and Tirole (2005b) for a discussion of our typology of license types), but there is not a consistent relationship between license strength and corporate contributions. The corporate contributions are more common when venture capitalists have funded a company that is focusing on the open source project, but this difference is

not statistically significant. Finally, consistent with the results regarding project size above, corporate contributions are more common in later versions of projects.

These measures are likely to be highly correlated, so it is natural to seek to understand their relative significance through a regression analysis. Table 5 presents such an analysis. We use as observations here each version of each project in the sample (to address heteroskedasticity concerns, we cluster the standard errors by project). In each case, the dependent variable is the share of corporate contributions, and the independent variables are those used in Table 4. (Both the code base and the growth rate are measured in millions of lines of code.) We employ an ordinary least squares (OLS) specification.

We estimate several variants of these regressions. Because there is a considerable degree of correlation between the size of the project and the rate of growth, we also present the analysis with only one of these measures. In addition, as noted above, the dynamics of projects that are spun off from a corporation or have a single major corporate sponsor may be fundamentally different. We thus estimated these regressions with and without these observations.

The regressions reveal that later versions of the project are consistently associated with greater corporate contributions. Moving from the 25th to the 75th percentile (from the fourth to the tenth version), is associated with a 12% increase in the corporate share, a substantial increase relative to the mean share of 32%. A larger and faster-growing code base is also associated with more contributions, but these relationships are less statistically robust. Finally, when we examine the subset of projects which were not spun off and without a major sponsor, venture capital-backing of a related firm is also associated with greater corporate contributions.

B. Examining the Sensitivity of Contributions to Growth

We now focus on the central prediction, and examine the sensitivity of the contributions of different classes of contributors to project growth. Our primary focus will be on understanding the relative magnitude of the coefficients on corporate contributors and hobbyists.

The first column of Table 6 presents the baseline analysis. We use as observations the contributions for each class of contributor to each version of each project in the sample. Thus, for a given version of the program, there will be five distinct observations. The dependent variable in each case is the number of contributions to that version by members of that class (e.g., hobbyists). We use as independent variables dummy variables for each of the five classes of contributors, interactions between each of the dummies and the growth of the project (measured in millions of lines of code), and a control for the number of months between the current and previous version. Because many of the observations are zero, we employ a Tobit specification.

The baseline regression reveals that corporate contributors are considerably more sensitive to project growth than other contributors. The coefficient for corporate contributors is four times that for hobbyists. The null hypothesis that two measures are identical is rejected at a high level of statistical significance.

Table 6 also presents three other regression results to address concerns about the baseline analysis. The reported regressions are intended to be representative. We did similar analyses of all the regressions reported in Tables 7 and 8 using these alternative approaches and found the results little changed.

One concern related to the panel nature of the data-set. The baseline specification did not explicitly address this feature of the data. We repeated the analysis, estimating OLS regressions with fixed and random effects. The results are similar, with the ratio of the sensitivity of corporate contributors to growth and that of hobbyists increasing to about 15.

Another concern involved the causality in the model. In particular, it is possible that there is a strong association between corporate contributions and project growth not because corporate contributions more responsive to contributions by others, but rather because the contributions by the corporate contributors are larger. We address this issue by using lagged project growth: that is, the change growth in the code base between the previous version (n-1) and the version prior to that (n-2). The results are little changed, suggesting that this worry is not a serious one.

In Table 7, we present the robustness of the results to a series of modifications of the specification:

- In the baseline analysis, we control for the fact that the calendar gap between different versions of the project is not always even by including the period between versions in months as an independent variable. We repeat the analysis in the first column, normalizing the dependent variable and the growth measure used to construct the independent variables by month. The results are little changed.⁸

⁸A more subtle issue is fact that the appearance of new versions of open source projects is not exogenous, but rather a choice by the project leaders. We repeat the analysis only using the subset of projects that appear through the sample period on a regular schedule (where presumably these endogeneity problems are less severe). The key results are essentially unchanged.

- As noted above, projects that are spun off from a corporation or with a major corporate sponsor may have fundamentally different dynamics than the others. In the second and third columns, we exclude such observations from the sample, and find the results to be slightly stronger than with the entire sample.
- The European open source community has been more ideological than its U.S. counterpart, efficiently organizing to fight against software patents and to pursue other open source causes. It might be thought that external motivations would be particularly strong here, and as a result, the difference between sensitivity of the contributions to growth between the hobbyists and corporate employees larger. The fourth column supports this suggestion.
- As noted above, the selection of projects not compiled in SourceForge was not random. It might be feared that the selection process introduced biases. We repeat the analysis, just using the SourceForge projects, and find a similar pattern as that in the entire sample.

C. Sensitivity of Contributions to Project Characteristics

In the final section of the paper, we seek to explore whether the relative sensitivity of corporate and individual contributors to the growth rate varies with the projects' characteristics. As delineated above, we anticipate that projects with greater commercial potential will attract more corporate interest, and this interest may be more sensitive to the commercial prospects of the project.

Table 8 presents the ratio of the sensitivity of corporate to hobbyist contributions to growth, dividing the observations by the characteristics of the project. The first two

columns reveal that the sensitivity to growth is greater for corporate contributors among larger projects: for projects above the median size, corporations are almost five times more sensitive to growth than hobbyists, while there is not significant difference between the two in those programs below the median size. When we compare these ratios in a Wald test, we find that the difference is statistically significant at the 5% confidence level.

We also divide the projects along some other dimensions. In two other cases, the differences in the sensitivity ratios are statistically significant. In projects with unrestrictive licenses (which presumably offer greater commercial opportunities), corporate contributors are more sensitive to performance, as is the case among faster-growing projects. (No significance differences in the ratios appear across the sub-samples when we divide the observations by venture capital activity or the version of the project.) Taken together, the results provide some support for the theoretical suggestions above.

6. Conclusions

This paper examines contributions to open source projects. We predict that the share of corporate contributions should be more sensitive to the growth of the project than those of the hobbyists and that this effect should be particularly strong when the commercial prospects of the project are favorable. We then test these ideas empirically using a panel data-set of activity in one hundred open source projects between 2001 and 2004. We find several patterns consistent with theoretical predictions.

The theory suggests a variety of other implications, which we hope to explore in future empirical work. One of the most intriguing of these relates to the concentration of

contributors and how this will evolve over time. In this analysis, we have focused on contributions by each class of contributor, but not exploited the very detailed data on the individual contributors we have compiled. In future work, we hope to exploit this information to test the theoretical suggestions.

References

Casadesus-Masanell, Ramon, and Pankaj Ghemawat, 2003, "Dynamic Mixed Duopoly: A Model Motivated by Linux vs. Windows," Strategy Unit Working Paper 04-012, Graduate School of Business Administration, Harvard University.

Gaudeul, Alexandre, 2004, "Competition between Open-Source and Proprietary Software: The (L^A)T_EX case Study," Unpublished working paper, Universities of Toulouse and Southampton.

Hann, Il-Horn, Jeff Roberts, Sandra Slaughter, and Roy Fielding, 2004, "An Empirical Analysis of Economic Returns to Open Source Participation," Unpublished working paper, Carnegie-Mellon University.

Haruvy, Ernan E., Fang Wu, and Sujoy Chakravarty, 2003, "Incentives for Developers' Contributions and Product Performance Metrics in Open Source Development: An Empirical Investigation," Unpublished working paper, University of Texas at Dallas.

Holmström, Bengt, 1999, "Managerial Incentive Problems: A Dynamic Perspective," *Review of Economic Studies*, 66, 169-182.

Johnson, Justin P., 2002, "Open Source Software: Private Provision of a Public Good," *Journal of Economics and Management Strategy*, 11, 637-662.

Lakhani, Karim, and Eric von Hippel, 2003, "How Open Source Software Works: 'Free' User-to-User Assistance," *Research Policy*, 32, 923-943.

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

Lerner, Josh, and Jean Tirole, 2005a, "The Economics of Technology Sharing: Open Source and Beyond," *Journal of Economic Perspectives*, 19, 99-120.

Lerner, Josh, and Jean Tirole, 2005b, "The Scope of Open Source Licensing," *Journal of Law, Economics, and Organization*, 21, 20-56.

Raymond, Eric, 1999, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Cambridge, O'Reilly.

Table 1: Project Characteristics

Sample:						
20 large projects						
78 randomly selected projects						
77 projects available from SourceForge						
98 total projects						
Started tracking:	05-2001					
Ended tracking:	07-2004					
	min	median			max	
Lines of Source Code ¹	1253 Wings 3D	81671 jEdit			4032921 Linux	
Absolute Change in Source Code ¹	-145395 AOLServer	18951 Licq			1628979 Linux	
Number of New Versions	1 Dev-C++ Imprints Kxicq KDE	8 Koffice BZFlag glibc			20 Wine	
License Type ²	Restrictive 73.80%			Highly Restrictive 51.40%		
For SourceForge Projects Only:						
Development Status	Planning 1.30%	Pre-Alpha 1.30%	Alpha 15.58%	Beta 37.66%	Production/ Stable 32.47%	Mature 11.69%
Intended Audience	Developers 51	End User 52	stem Administra 28	Other 16		
Operating System	Operating System Independent 25	Linux 44	POSIX 30	Microsoft 22	MAC 6	
Programming Language	C/C++ 59	Perl 10	Python 9	Java 15	Other Language 15	
Topic ³	Diversions 28	Office 16	Education 3	Communication 5	System Utilities 31	

Notes:

1. Measured at the end of the sample

2. BSD is an unrestrictive license, LGPL is restrictive, and GPL is highly restrictive. Three projects changed their license during our sample period: Sendmail, PureFTPd, Wine

3. A project can only be in one category

Table 2: Characteristics of Contributions

Panel A: Characteristics By Project

	min	median	max
Number of Contributors ¹	1 CsvJdbc EverQuest JFS	67 Miranda ICQ client Gstreamer	3521 Linux
Absolute Growth in Contributors	-343 JFS	16 Common C++ Libraries Gstreamer Jext	1174 Linux
% Growth in Contributors	-99.70% JFS	35.9% Gabber	4200% PPTP Client
Number of Contributions ¹	2 JFS	374 Cluster Infrastructure	52607 GCC
Absolute Growth in Contributions	-1208 XFree86	80 ROX Desktop	24611 GCC
% Growth in Contributions	-99.50% JFS	49.80% Licq	5800% AWStats
Share of contributions of top 1 ¹	1.80% Linux	30.45% Gabber	100.00% 5 projects
Share of contributions of top 5 ¹	7.30% Linux	55.35% Gnumeric	100.00% 7 projects
Share of contributions of top 20 ¹	18.35% Linux	91.29% jEdit	100.00% 23 projects
10% of contributions are done by ¹	1	1	7
25% of contributions are done by ¹	1	1	35
50% of contributions are done by ¹	1	3.5	193
Concentration of Internal Contributors ^{1,2}	0	0	70.90% JXTA
Concentration of Internal Contributions ^{1,2}	0	0	93.44% Zsnes

Panel B: Aggregate Characteristics

<i>Contributors</i>		Commercial	Nonprofit	US	Network	International	Other
	total		Organization	Educational			
all	27,096	31.7%	14.8%	8.7%	8.6%	36.2%	0.1%
first	21,792	29.8%	11.9%	11.6%	6.7%	35.0%	4.9%
last	26,811	30.1%	14.0%	9.5%	7.7%	34.5%	4.2%
<i>Contributions</i>		Commercial	Nonprofit	US	Network	International	Other
	total		Organization	Educational			
all	4,176,101	33.6%	25.6%	10.0%	5.4%	24.3%	1.1%
first	150,642	33.6%	20.4%	12.7%	5.2%	28.1%	0.1%
last	260,258	35.1%	24.0%	10.0%	5.9%	24.9%	0.1%

Notes:

1. Measured at the end of the sample
2. 27 projects identified with internal contributors

Table 3: Composition of Contributors

Rank	Top-level domain	Definition	Contributors	Percent	Contributions
1	COM	Commercial	8077	30.13%	91348
2	ORG	Nonprofit Organizatic	3742	13.96%	62539
3	EDU	US Educational	2550	9.51%	25983
4	DE	Germany	2016	7.52%	16152
5	NET	Network	2067	7.71%	15204
6	UK	United Kingdom	767	2.86%	6248
7	FR	France	498	1.86%	4130
8	US	United States	368	1.37%	3092
9	CZ	Czech Republic	222	0.83%	3000
10	NL	Netherlands	411	1.53%	2652
11	AU	Australia	520	1.94%	2576
12	JP	Japan	408	1.52%	2393
13	CA	Canada	355	1.32%	2078
14	SE	Sweden	335	1.25%	1922
15	NO	Norway	231	0.86%	1871
16	IT	Italy	192	0.72%	1837
17	DK	Denmark	208	0.78%	1649
18	BE	Belgium	129	0.48%	1502
19	CH	Switzerland	170	0.63%	1443
20	FI	Finland	259	0.97%	1374
21	RU	Russian Federation	221	0.82%	1107
22	IL	Israel	68	0.25%	1068
23	BR	Brazil	169	0.63%	961
24	HU	Hungary	113	0.42%	872
25	PL	Poland	146	0.54%	813
26	AT	Austria	159	0.59%	790
27	NZ	New Zealand (Aotear	83	0.31%	763
28	GOV	US Government	213	0.79%	515
29	MX	Mexico	32	0.12%	398
30	ES	Spain	106	0.40%	319
31	CX	Christmas Island	25	0.09%	213
32	PT	Portugal	41	0.15%	212
33	MIL	US Military	79	0.29%	204
34	TW	Taiwan	80	0.30%	190
35	HR	Croatia (Hrvatska)	45	0.17%	188
36	IS	Iceland	19	0.07%	188
37	IE	Ireland	35	0.13%	182
38	ZA	South Africa	38	0.14%	148
39	EE	Estonia	51	0.19%	135
40	GR	Greece	30	0.11%	128

Table 4: Distribution of Corporate Contributors As a Share of All Contributors

Size of Code Base		Growth of Code Base		License Type		Venture Backing		Version	
Smallest size quartile	21.4%	Smallest growth quartile	29.9%	Unrestrictive licenses	32.0%	Venture-backed projects	35.0%	Less than version 4	5.5%
Mid-small size quartile	22.2%	Mid-small growth quartile	26.3%	Restrictive licenses	37.1%	Non-venture backed	31.6%	Version 4 to 6	24.8%
Mid-large size quartile	33.1%	Mid-large growth quartile	26.6%	Highly restrictive licenses	29.0%			Version 7 to 11	38.4%
Largest size quartile	44.2%	Largest growth quartile	43.3%					More than version 12	43.9%
p-Value, F- (or t-)test	0.000		0.000		0.093		0.398		0.000

Table 5: Regression Analysis of Corporate Contributors

	Dependent Variable: Corporate Contributors as a Share of All Contributors							
Size of code base	0.07	<i>1.88</i> **	0.04	<i>1.60</i>	0.07	<i>1.64</i> **	0.02	<i>0.84</i>
Growth of code base					0.14	<i>0.62</i>	0.63	<i>1.71</i> *
Restrictiveness index	-0.01	<i>0.29</i>	0.01	<i>0.31</i>	-0.01	<i>0.32</i>	0.01	<i>0.20</i>
Venture-backed project	-0.02	<i>0.38</i>	0.08	<i>1.84</i> *	-0.02	<i>0.41</i>	0.07	<i>1.85</i> *
Version	0.02	<i>4.48</i> ***	0.02	<i>2.80</i> ***	0.02	<i>4.53</i> ***	0.02	<i>2.87</i> ***
Constant	0.12	<i>1.95</i> *	0.08	<i>1.17</i>	0.12	<i>1.94</i> *	0.08	<i>1.69</i>
Includes spin-offs/sole sponsors?	Y		N		Y		N	
Number of observations	385		294		385		294	
F-statistic	15.45		14.05		9.04		11.68	
p-Value	0.000		0.000		0.000		0.000	
R-squared	0.14		0.12		0.14		0.13	

Notes:

Ordinary least squares regressions.

Each version of each project in the sample is an observation.

Heteroskedastic-consistent standard errors (clustered by project) in italics.

Size and growth of code base is in millions of lines.

***=statistically significant at the 1% confidence level; **=5%; *=10%.

Table 6: Sensitivity of Contributors to Project Growth

	Dependent Variable: Number of New Contributions by Contributor Type							
	<i>Tobit Specification</i>		<i>Fixed-Effect Specification</i>		<i>Random-Effects Specification</i>		<i>Tobit Specification--With Lagged Growth</i>	
Project growth * Corporate contributors	137.2	<i>44.13 ***</i>	121.5	<i>22.36 ***</i>	120.8	<i>22.39 ***</i>	156.0	<i>9.49 ***</i>
Project growth * "Hobbyists"	34.2	<i>3.76 ***</i>	8.5	<i>1.57</i>	7.8	<i>1.44</i>	48.6	<i>2.83 ***</i>
Project growth * International contributors	48.8	<i>5.54 ***</i>	31.8	<i>5.85 ***</i>	31.0	<i>5.75 ***</i>	22.0	<i>3.12 ***</i>
Project growth * ".Net" contributors	26.4	<i>2.90 ***</i>	7.5	<i>1.38</i>	6.7	<i>1.25</i>	-3.9	<i>0.07</i>
Project growth * ".Org" contributors	50.9	<i>5.87 ***</i>	34.9	<i>6.42 ***</i>	34.1	<i>6.32 ***</i>	95.6	<i>3.20 ***</i>
Months between project versions	-0.7	<i>5.74 ***</i>	0.2	<i>1.92 *</i>	0.2	<i>1,98</i>	0.1	<i>1.13</i>
Dummy variable for contributor type?	Y		Y		Y		Y	
Number of observations	2244		2244		2244		2061	
Log likelihood or F- or χ^2 -statistic	-5150.7		14.05		938.24		-5192.3	
p-Value			0.000		0.000			
R-squared			0.48		0.27			
Ratio, Corporate/"Hobbyist"	4.01		14.29		15.49		3.21	
t-Test of Equality	0.000		0.000		0.000		0.000	

Notes:

Contributions by each class of participants in each version of each project in the sample are observations.

Standard errors in italics.

Project growth is in millions of lines code.

***=statistically significant at the 1% confidence level; **=5%; *=10%.

Table 7: Robustness of Key Result to Changes in Specification

	<u>Normalizing All Variables by Months Between Versions</u>	<u>Eliminating Spun- Off Projects</u>	<u>Eliminating Spun-Off and Sole-Sponsored Projects</u>	<u>Using European "Hobbyists" Only</u>	<u>Using End- User Oriented Projects Only</u>
Ratio, Corporate/"Hobbyist"	4.90	4.34	4.34	4.51	4.34
t-Test of Equality of Coefficients	0.000	0.000	0.000	0.000	0.000

Note:

All regressions employ Tobit specifications as in first regression in Table 3.

Contributions by each class of participants in each version of each project in the sample are observations.

Table 8: Sensitivity of Contributors to Project Growth, by Project Type

	Size of Code Base		Growth of Code Base		Venture Backing		License Type			Version	
	<i>Below Med.</i>	<i>Above Med.</i>	<i>Below Med.</i>	<i>Above Med.</i>	<i>Yes</i>	<i>No</i>	<i>Unrestrictive</i>	<i>Restrictive</i>	<i>Highly Restrictive</i>	<i><7</i>	<i>≥7</i>
Ratio, Corporate/"Hobbyist"	0.69	4.87	0.92	5.23	5.27	3.70	NA	6.52	4.58	10.41	2.05
t-Test of Equality of Coefficients	0.372	0.000	0.874	0.000	0.000	0.000	0.300	0.423	0.000	0.000	0.003
t-Test of Equality of Ratios in Sub-Samples	0.027		0.055		0.572		0.002			0.136	

Notes:

All regressions employ Tobit specifications as in first regression in Table 3.

Contributions by each class of participants in each version of each project in the sample are observations.

NA=In this case, hobbyists actually have a negative sensitivity to project growth, so a meaningful ratio cannot be computed.

Appendix: List of Projects Tracked

Project Name	# of Releases	In SourceForge?	Lines of Source Code ¹	Internal e-mail
AbiWord	17	Y	189,752	abisource.com
AFPL Ghostscript	7	Y	235,890	alladin.com, ghostscript.com, ghostgum.com.au, ghostview@cs.wisc.edu
AOLServer	9	Y	189,310	
Apache (httpd)	14	N	124,636	apache.org
Audacity	6	Y	46,395	
AWStats	12	Y	3,156	
Aztec 3D Modeller	5	Y	49,273	
Battletech MUX	2	Y	104,348	
BIND	7	N	198,652	
BlackNova Traders	5	Y	18,411	blacknova.net
Bochs x86 PC emulator	9	N	73,343	
BZFlag	8	Y	54,489	bzflag.de
Cajun	5	Y	3,619	
Cal3D	5	Y	13,391	
Celestia	7	Y	26,687	
Cluster Infrastructure for Linux	4	Y	.	
Common C++ Libraries	11	N	20,230	
Cronos II	4	Y	31,462	
Crystal Space 3D engine	5	Y	250,217	
CsvJdbc	3	Y	.	
Dev-C++	1	Y	60,009	
Digikam	7	Y	13,941	
DOOM Legacy	7	Y	112,082	newdoom.com
Dprobes	7	N	2,564	
Eclipse	13	N	14,841	
Emacs	4	N	531,959	
EverQuest Server Emulator	11	Y	16,561	
Exult	9	Y	66,792	
Fast Light Tool Kit	10	Y	39,047	ftlk-bugs@easysw.com, fltk.org
Firewall Builder	12	Y	22,637	fwbuilder.org
Fluxbox	15	Y	24,540	fluxbox.org
Freenet Project	6	Y	48,245	freenetproject.org
FreeTTS	5	Y	4,168	
Gabber	5	Y	35,857	jabber.org
Gaim	17	Y	79,362	
GCC	14	N	1,174,888	
glibc	8	N	632,401	
GNOME News Applet	4	Y	10,432	
Gnumeric	19	N	151,722	
GriD Engine	9	N	368,497	
Gstreamer	11	Y	45,619	
Gutenbrowser	2	Y	7,460	
HP Inkjet Linux Driver	10	N	19,887	hp.com
HP OfficeJet Linux Driver	3	N	17,074	
HSQL Database Engine	5	N	14,723	
Ickle	4	Y	28,259	
Imprints	1	Y	16,939	
Jboss Server (jboss.org)	17	N	35,121	jboss.org
jEdit	6	Y	49,839	jedit.org
Jext	10	Y	51,671	
JFS	12	N	15,396	
Jikes	10	N	181,934	
JXTA	5	N	103,449	jxta.org

KDE	1	N	1,057,806	kde.org
KOffice	8	N	333,232	
Kxircq	1	Y	39,948	
Licq	5	Y	57,313	
Linux	13	N	2,403,942	redhat.com, valinux.com
Linux NTFS support	10	Y	17,708	
Linux PCMCIA Card Services	10	Y	68,467	
Log library for C++	6	Y	9,127	
Lopster	3	Y	45,079	lopster.irc
lpr	3	Y	46,304	
Mesa3D	10	Y	202,640	
Miranda ICQ client	9	Y	16,983	
Mozilla	18	N	2,153,933	
MySQL + MySQL MAX	17	N	175,830	mysql.com
Netatalk	10	Y	50,877	
Netscript	3	Y	4,587	
NetTime	2	Y	3,991	
Numerical Python	10	Y	59,122	
OGRE	13	Y	70,530	
OMNI Printer Driver	10	Y	74,687	
Open Source Database Benchmark	3	Y	5,350	
OpenOffice	14	N	2,912,015	
ORP	3	Y	124,480	
PHP	15	N	209,003	
POI/POI Serialization Project	2	Y	49,957	
PostgreSQL	14	N	312,995	postgresql.org
PPTP Client	6	N	583	
Pure-FTPd	11	Y	13,440	pureftpd.org
Python	13	N	275,555	python.net, python.org, pythonware.com
QuakeForge	4	Y	78,299	quakeforge.net
ROX Desktop	13	Y	24,598	
Samba	12	N	163,876	samba.org
Sendmail	15	N	65,550	
Single System Image Clusters for Lin	10	Y	76,167	
Slash	4	Y	31,054	
Small device C compiler	3	Y	156,117	
Solaris-compatible Thread Library	4	Y	6,091	
Tapestry Web Components	11	Y	22,948	
UOX3	2	Y	83,239	
Visual Component Framework	12	Y	237,295	
Wine	20	N	472,744	winehq.com
Wings 3D	16	Y	275	
XFree86	3	N	1,932,631	
Zsh	7	Y	76,544	zsh.org
ZSNES	5	Y	114,773	zsnes.com

¹ In latest project release