The Dynamics of Open-Source Contributors

By Josh Lerner, Parag A. Pathak, and Jean Tirole*

There are substantial differences between open-source projects and traditional innovative efforts in private firms. Private firms usually pay their workers, direct and manage their efforts, and control the output and intellectual property created. In an open-source project, however, a body of original material is made publicly available for others to use, under certain conditions. Contributions to open-source projects are made by a diverse array of individual contributors, and for-profit corporations, who must often agree to make enhancements to the original material widely available for nominal cost.

This paper empirically examines the dynamics of contributions to open-source software projects. We show that the share of corporate contributions in a sample of approximately 100 open-source projects between 2001 and 2004 is greater in larger and growing projects.

I. Background[1]

The decision to contribute without pay to freely available software may seem mysterious to economists. The unpaid programmer working on an open-source software development project, however, 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 earnings that could come from 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 drop and the student's progress toward a degree may slow.

Several short- or long-run benefits may counter these costs (Lerner and Tirole, 2002). 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. 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, future access to the venture capital market, and ego gratification from peer recognition. 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: (a) the more visible the performance to the relevant audience (peers, labor market, and venture capital community); (b) the higher the impact of effort on performance; and (c) the more informative the performance about talent (for example, Bengt 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

* Lerner: Harvard Business School, Harvard University, Rock Center, Boston, MA 02163, and NBER (e-mail: josh@hbs.edu); Pathak: Harvard Business School, Harvard University, Baker Library 420E, Boston, MA 02163 (e-mail: ppathak@fas.harvard.edu); Tirole: Universite des Sciences Sociales, Manufacture des Tabacs, Aile Jean-Jacques Laffont, Accueil MF 404, F-31000, Toulouse, France, and Massachusetts Institute of Technology (e-mail: tirole@cict.fr). We thank Vakha Elmurzev, Lee Gao, James Hunter, Payal Loungani, Susanna Kim, Qicheng Ma, and John Sheridan for research assistance. Tim Bresnahan, participants in the 2006 ASSA annual meeting, the Toulouse Network on Information Technology's Fall 2005 meeting, and a number of practitioners provided helpful comments. Jeff Bates was extremely helpful with SourceForge access. We thank Harvard Business School, the National Science Foundation, and the Toulouse Network on Information Technology for financial support.

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

the possibility of multiple equilibria. The same project may attract few programmers because programmers expect other programmers will not be interested; or it may flourish as programmers (rationally) have faith in the project.

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. 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 with the expectation that they will capture a return from a proprietary project.

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 that 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. Firms may also be able to learn about potential employees when their staff interacts with open-source programmers. 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 that 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 of 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 akin 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.

II. The Sample

We built a panel dataset of the contributors to approximately 100 open-source projects (for full details on the dataset, see Lerner et al., 2006). These projects are stratified to overrepresent the largest open-source projects. We extract

TABLE 1—PROJECT CHARACTERISTICS

Sample:			
20 large projects	Started tracking:		05-2001
78 randomly selected projects	Ended tracking:		07-2004
98 total projects			
	min	median	max
Lines of source code*	1,253 Wings 3D	81,671 jEdit	4,032,921 Linux
Absolute change in source code*	−145,395 AOLServer	18,951 Licq	1,628,979 Linux
Number of new versions	1 Dev-C++ Imprints Kxicq KDE	8 Koffice BZFlag glibc	20 Wine

	Restrictive	Highly restrictive
License type**	74%	51%

* Measured at the end of the sample.

** 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. We take the license post-change.

the contributors to the project in each new official version of the program that has been released, using a variety of text editing tools.

Table 1 summarizes the projects and highlights that they differ considerably in their size and other characteristics. Open-source projects periodically introduce new versions. The number of versions introduced between the beginning of data collection and July 2004 varies between one and 20.[2] We obtained information on the projects from SourceForge (the leading on-line depository of open-source projects), press searches, and project Web sites. Key information includes the type of license of the project, whether venture capitalists funded the company, and whether a corporation released some of its code as an open-source project.

For each project, we opened the Tape Archive (known as "tarball") to count the number of distinct references to each individual contributor. The archive preserves information such as user and group permissions, dates, and directory structures. 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."[3] This point is also emphasized by Eric Raymond (1999), who points out "surreptitiously filing someone's name off 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. Each project release was then associated with a set of e-mails that appeared in the archive.[4]

We aimed to distinguish individuals who were contributing code on their own behalf from those doing so as part of their employment. Our approach divided the contributors into five classes based on their e-mail addresses: corporate employees, individual hobbyists,

[2] We 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.

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

[4] The database went through an extensive cleaning process to remove invalid e-mail addresses and to deal with situations where there were two e-mail addresses for the same individual. Examples of the decisions made are in Lerner et al. (2006).

TABLE 2—CHARACTERISTICS OF CONTRIBUTIONS

	min	median	max
Number of contributors*	1 CsvJdbc EverQuest JFS	67 Miranda ICQ client Gstreamer	3,521 Linux
Absolute growth in contributors	−343 JFS	16 Common C++ Libraries Gsteamer Jext	1,174 Linux
% growth in contributors	−100 JFS	36 Gabber	4200 PPTP Client
Number of contributions*	2 JFS	374 Cluster infrastructure	52,607 GCC
Absolute growth in contributions	−1,208 XFree86	80 ROX Desktop	24,611 GCC
% growth in contributions	−100 JFS	50 Licq	5800 AWStats

* Measured at the end of the sample

three classes of otherwise "other" contributors, unidentified international contributors, and those from organizations with top-level domains (TLDs) denoted ".org" and ".net," which frequently indicate nonprofit 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 (ISPs), 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.[5] 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. Table 2 presents details of characteristics of contributors.

III. Analysis of Project Contributions

Our initial analysis seeks to understand the distribution of contributions to open-source project by class of contributor, focusing on contributions by corporations and "hobbyists." Table 3 presents some breakdowns, 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.[6] The table also presents the result of *F*- and *t*-tests of the significance of the reported differences.

The table shows the share of corporate contributions is twice as large in the largest quartile of projects as in the smallest quartile. 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

[5] One 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 separately. With this further breakout of the corporate sample, the qualitative results are similar.

[6] Results looking at the ratio of contributions by corporate contributors and hobbyists generate similar results.

TABLE 3—DISTRIBUTION OF CORPORATE CONTRIBUTORS AS A SHARE OF ALL CONTRIBUTORS

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

projects with the most restrictive licenses (see Lerner and Tirole, 2005b, for a discussion of our typology of license types), but there is no consistent relationship between license strength and corporate contributions. 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, corporate contributions are more common in later versions of projects.

IV. Conclusions

This paper presents only the beginnings of understanding cross-sectional and time-series patterns of contributions to open-source projects. In the approximately 100 software projects we track from 2001 to 2004, we have shown that the share of corporate contributions is much larger in large and growing projects. In a companion paper, we develop a theoretical rationale for these patterns and explore them in more depth.

REFERENCES

Holmström, Bengt. "Managerial Incentive Problems: A Dynamic Perspective." *Review of Economic Studies*, 1999, *66*(1), pp. 169–82.

Lerner, Josh and Tirole, Jean. "Some Simple Economics of Open Source." *Journal of Industrial Economics*, 2002, *50*(2), pp. 197–234.

Lerner, Josh and Tirole, Jean. "The Economics of Technology Sharing: Open Source and Beyond." *Journal of Economic Perspectives*, 2005a, *19*(2), pp. 99–120.

Lerner, Josh and Tirole, Jean. "The Scope of Open Source Licensing." *Journal of Law, Economics, and Organization*, 2005b, *21*(1), pp. 20–56.

Lerner, Josh; Pathak, Parag A. and Tirole, Jean. "Open Source Contributors." Unpublished Paper, 2006.

Raymond, Eric. *The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary*. Sebastopol, CA: O'Reilly & Associates, 1999.