Exploring the Open Source Software Bazaar

Balaji Rajagopalan*
School of Business Administration
Oakland University
Rochester, MI  48309
(248) 370-4958
rajagopa@oakland.edu

Barry L. Bayus
Kenan-Flagler Business School
University of North Carolina
Chapel Hill, NC  27599
(919) 962-3210
Barry_Bayus@UNC.edu

August 2004

* Both authors contributed equally.
Exploring the Open Source Software Bazaar

ABSTRACT

The purpose of this paper is to explore two of Raymond’s key open source bazaar principles. We empirically examine the relationships between project community size (“eyeballs”) and development activity, and between development activity and product adoption. We find strong evidence to support the premise that “developer eyeballs” are positively related to development activity. Based on a proportional hazard analysis of time to adoption takeoff, we also find that product development activity is significantly related to the speed of product adoption. We interpret these results as supporting some key principles of the open source bazaar: (1) attracting a large developer base is important to further product development, and (2) the early market evolution and acceptance of open source products is driven by product development activity. Contrary to the tenets of the bazaar model, however, we find that “user eyeballs” do not significantly contribute to increased development activity. In addition, we find evidence suggesting that product success does not in turn drive additional product development. Thus, our results also suggest that the bazaar community development model involving developers and users originally proposed by Raymond needs revision for the more typical open source software project.
1. Introduction

The open source software movement is largely comprised of highly skilled professional developers with a commitment to the philosophy of “open source code” (David, Waterman and Arora 2003). These developers collaborate over the Internet, generally voluntarily developing software during their own spare time. From its humble beginnings by a few dedicated programmers, the open source software movement has now grown to over 85,000 projects and 890,000 registered users (a count as of August 2004 from sourceforge.net, the largest repository of open source projects on the Internet). For example, open source software is pervasive in the web server domain with Apache leading the way in having the largest market share (almost 70% according Netcraft’s Web Server survey as of August 2004), and the Linux operating system is emerging as a strong contender in the desktop category. The huge success of open source products like Apache and Linux in attracting a large user base has captured the attention of the business world, as well as academic researchers (e.g., see the reviews in Krishnamurthy 2003; von Hippel and von Krogh 2003; von Krogh and von Hippel 2003). Not surprisingly, one important topic is concerned with better understanding the evolution open source software systems and its underlying product development process (e.g., Kemerer and Slaughter 1999; Godfrey and Tu 2000; Krishnamurthy 2002; Healy and Schussman 2003; Scacchi 2003; Paulson, Succi and Ebertein 2004).

The open source development process has been characterized as a bazaar, a term coined by Raymond (1998). The bazaar model of product development is based on two key principles: (1) “given enough eyeballs, all bugs are shallow,” and (2) “release often and release early” (Raymond 1998). This approach is in direct contrast to the
traditional software development process which Raymond characterizes as a *cathedral* (i.e., products are developed using a centralized approach, designs are carefully drafted, and there are no beta prototype releases until all the major bugs are resolved; Raymond 1998).

Originally constructed based on a qualitative examination of Linus Torvalds’ development of the Linux operating system, Raymond’s community development model has been supported by other case studies of large successful systems including Linux (e.g., Godfrey and Tu 2000), Apache (e.g., Mockus, Fielding and Herbsleb 2000; Lakhani and von Hippel 2003), Mozilla (e.g., Mockus, Fielding and Herbsleb 2002), and GNOME (e.g., Koch and Schneider 2000). Nevertheless, research to date is equivocal on whether the bazaar development principles apply to more “typical” open source software projects that never achieve the kind of market success generally associated with these larger projects (e.g., Hunt and Johnson 2002; Krishnamurthy 2002; Healy and Schussman 2003; Koch 2004). For example, based on a study of thousands of projects available from the Sourceforge online database, Healy and Schussman (2003) find that the typical open source software project has one developer, no discussion or bug reports, and is not downloaded by anyone. Krishnamurthy (2002) reports similar results, proposing that the development of open source software is more consistent with a *lone developer* (or cave) model. However, these conclusions are based on simple descriptive statistics and the observation that development effort and project activity across open source software projects is highly skewed (i.e., many projects have few developers and little or no activity, while only a few have a lot of development interest
and activity). We are unaware of any empirical studies providing multivariate statistical support for (or against) the key bazaar development principles.

The purpose of this paper is to further explore the open source software bazaar. Consistent with Raymond’s key bazaar principles, our primary interest is in whether there is a statistical relationship between project community size (“eyeballs”) and development activity, and between development activity and product adoption. We empirically examine these relationships in a random sample of several hundred “typical” open source products hosted on Sourceforge. In agreement with the bazaar model, we find strong evidence that “developer eyeballs” are positively related to development activity (i.e., bug fixes, bug reports, support requests, commits, version releases). Based on a proportional hazard analysis, we also find that product development activity (e.g., rate of version releases and timing of first version release) is significantly related to the speed of product adoption. We interpret these results as supporting some key principles of the open source bazaar: (1) attracting a large developer base is important to further product development, and (2) the early market evolution and acceptance of open source products is driven by product development activity. Contrary to the tenets of the bazaar model, however, we find that “user eyeballs” do not significantly contribute to increased development activity. In addition, we find evidence suggesting that product success does not in turn drive additional product development. Thus, our results also suggest that the bazaar community development model involving developers and users originally proposed by Raymond (1998) needs revision for the more typical open source software project. In particular, it appears that users do not always play a critical role in
development during the early evolution of a project and development activities drop off once downloads dramatically increase.

The remainder of this paper is organized as follows. In the next section, we describe the open source in more detail and explain the basis for our research model. In § 3 we describe the data available for our analyses, present the empirical analysis and discuss our findings. We then conclude with some directions for future research in § 4.

2. Conceptual Framework

The underlying conceptual model that guides our study is shown in Figure 1. In line with the product and software development literatures, we expect that: (1) project community size (indicating development efforts) is positively related to development activity, and (2) development activity (and its associated product improvements) is positively related to the speed of product adoption (e.g., Clark and Wheelwright 1994; Cusumano and Selby 1996; Krishnan and Ulrich 2001; MacCormack, Verganti and Lansiti 2001; Garcia 2004; Wynn 2004). We note that this basic framework is consistent with results reported in the economics and technology literatures that new innovations typically begin in a primitive form and the early stages of new market growth focuses on continual product improvement (e.g., Klepper 1997; Agarwal and Bayus 2002; Wynn 2004). In addition, several studies demonstrate that product improvements, relative to process improvements, are emphasized in the early stages of a new market (e.g., Utterback 1994; Klepper 1997).

[insert Figure 1 about here]
In the context of open source software, this basic framework is consistent with Raymond’s (1998) bazaar model of product development. In particular, the ability to parallelize the debugging process is touted as a primary benefit of a large community. According to Linus’ law, given a sufficiently large group of developers and beta-testers (i.e., users), bugs will be found and fixed rapidly (Raymond 1998). In other words, developer and user “eyeballs” are expected to be positively associated with product development activity (e.g., bug reports, bug fixes, support requests, version releases).

Raymond (1998) goes on to state that an important project activity is releasing new versions early and often. New version releases enhance the debugging process, and thus lead to improved software products. Following the product development literature (e.g., Krishnan and Ulrich 2001; Agarwal and Bayus 2002), we expect that open source development activity is positively related to the speed of product adoption.

Importantly, this conceptual model and its associated relationships have not been empirically tested in the open source software environment (Garcia 2004). Instead, researchers have focused their attention on the motivations of programmers to voluntarily contribute to open source projects (e.g., Raymond, 1998; Cohendet, Creplet and Dupouet 2001; Hars and Ou 2001; Kasper 2001; Tuomi 2001; Lakhani and Wolf 2003; Rossi 2004), how the open source innovation process works (e.g., Mockus, et al. 1997; Mockus, Fielding and Herbsleb, 2000; Godfrey and Tu 2001; Lerner and Tirole, 2002; Jensen and Scacchi 2004a, Jensen and Scacchi 2004b; Paulson, Succi and Eberlein 2004, Zhao and Deek 2004) and the competitive dynamics associated with open source software products (e.g., Bonaccrossi and Rossi 2003; Dedrick and West 2003; Overby, Bharadwaj and Bharadwaj 2004).
To test the relationships in our conceptual model, we statistically analyze the cross-sectional variation across a large sample of “typical” open source projects. Based on our discussion in this section, we have three primary hypotheses: (1) projects with many developer eyeballs have more development activity than projects with few developer eyeballs; (2) projects with many user eyeballs have more development activity than projects with few user eyeballs; and (3) projects with versions that are released early and often have quicker product adoption than projects with little development activity. We empirically examine each of these hypotheses in the next section.

3. The Empirical Study

3.1 Data

Rather than emphasize only the larger, well-known open source projects, we study the relationships between project community size and development activity, as well as development activity and product adoption, using a random sample of open source projects hosted by Sourceforge. In addition to being the largest host for open source projects, Sourceforge is also a rich source of data on the open source development process (e.g., bugs, support requests, version release) and hence, is an excellent data source for researchers (Garcia 2004). Not surprisingly, other researchers are also drawing on this data source (e.g., Chengalur-Smith and Sidorova 2003; Healy and Schussman 2003; Xu and Madey 2004). Data for our study was

---

1 Freshmeat.net is another large host of open source applications with over 34,000 projects and 280,000 registered users as of August 2004. However, data for the variables of interest to us are not available from this data source.
collected from sourceforge.net in December 2003. In general, we exercised caution in acquiring, cleaning and analyzing data (see Howison and Crowston 2004).

As suggested by Garcia (2004), we used a web crawler to generate a comprehensive list of projects from the six largest project categories listed on Sourceforge (this comprised a total of 17,035 projects). It is important to note that assignments to project categories are not mutually exclusive. For the purposes of this study, the first category listed was chosen. From these projects, we randomly selected 1,000 projects to be included in our study. With the help of a webbot, data on project characteristics and development activity was then collected for these projects. Since fifteen projects had to be dropped for lack information or being duplicates of other projects in the set, the sample size for our study is 985 projects.

Our sample includes projects that are in different stages of development: 21% are in the planning phase, 14% are in the pre-alpha phase, 16% are in the alpha phase, 26% are in the beta phase, 20% are in the production/stable category, and 1.5% are in the mature phase of development (only three projects in our sample were listed as inactive). In terms of project age (as of December 2003), more than three-quarters of the projects are less than two years old and none are older than four years. The basic characteristics of our sample are similar to other studies using Sourceforge data (e.g., see Healy and Schussman 2003).

As shown in Table 1, about half of our sample has only one developer and one project manager. Additionally, almost half of the open source projects in our sample have more than one developer (Table 1: 22.2% + 26.4%). As reported in Table 2, a

---

2 Since we do not draw any specific conclusions based on the category, this has little impact on our analysis and conclusions.
majority of the projects in our sample had at least one version release and one commit. At the same time, more than half of our sampled projects have no bug reports, no bug fixes, and no support requests. We note that the highly skewed nature of these measures is consistent with the findings reported by other researchers (e.g., Hunt and Johnson 2002; Krishnamurthy 2002; Healy and Schussman 2003; Koch 2004).

[insert Tables 1 and 2 about here]

3.2 Developer Eyeballs and Development Activity

To empirically examine Linus’ law, we now explore the relationship between developer eyeballs and development activity. Our measure of developer eyeballs is based on the number of developers participating in a project\(^3\). Given the existence of a highly skewed distribution of development effort across open source projects, we define two categories to facilitate our analysis and interpretation\(^4\): one developer and more than one developer.

Software development activity is typically characterized as a function of both new development and the fixing of bugs (Garcia 2004). In line with this, product development activity is measured as the number of bugs fixed, bug reports, version releases, support requests, and commits to make future coding changes. Three of the measures – bugs fixed, version releases, and commits – are directly indicative of product development activities. Although the other two dimensions – support requests and bug reports – most likely originate from the user community, they are likely to spur

\(^3\) For the projects in our sample, the number of developers and project managers working on each project as of December 2003 was collected. To check for stability over time, we revisited this measure six months later. Consistent with Krishnamurthy (2002), we find that the developer count is relatively stable over the life of a project. Of the 985 projects in our study, less than fifteen had any major count change.

\(^4\) Due to the highly skewed nature of these data, regression (correlation) analysis is inappropriate (e.g., Garcia 2004). Thus, we will rely on statistical tests of two proportions.
development and hence, serve as indirect indicators of development activity. Again, due to the highly skewed nature of these measures across open source projects, we define two categories for each measure: zero and at least one.

A cross-tabulation of developer eyeballs and development activity is in Table 3. It is clear that projects with more than one developer are significantly more likely to have bug fixes, bug reports, support requests, commits, and version releases. For example, 29.1% of open source projects with one developer had at least one bug report, whereas 52.2% of projects with more than one developer had at least one bug report (this difference is significant at better than 0.01). We interpret this analysis to strongly indicate that a larger developer base is associated with increased levels of product development activity, supporting the bazaar development principle that more developer eyeballs are associated with higher development activity.

3.3 User Eyeballs and Development Activity

Unfortunately, direct measures of the number of users for an open source product are generally unavailable (e.g., Garcia 2004). Consequently, we use the cumulative number of downloads as a proxy for users (see also Garcia 2004; Wynn 2004). We recognize that one user may download multiple copies and/or pass along a downloaded version to several other users and hence, downloads has its limitations.

---

5 Although not reported here, we also find that open source projects with more than one project manager are significantly more likely to have bug reports, bug fixes, support requests and commits (but not version releases).

6 We note that this does not really contradict prior findings that a small number of developers (especially in large project communities) contribute a large fraction of the code (e.g., Ghosh and Prakash 2000 find that 10% of the contributors in their study accounted for over 70% of the code). Indeed, CVS data on lines of code for the projects in our sample (with at least 6 months of LOC information; n=349) indicates that more than 20% of the open source projects with one project manager and one developer were able to generate 100,000 or more lines of code. This is not statistically different than projects with more than developer. Details are available from the authors.
(e.g., see Howison and Crowston 2004). But, we believe these problems are less severe for small or medium size projects (the type of which are “typical” of open source projects on Sourceforge and our sample) where alternate distribution channels are not the mainstay.

Examples of the cumulative download pattern for four open source projects in our sample are shown in Figure 2. These patterns exhibit the well-known “takeoff” phenomenon, i.e., downloads are very low (if not zero) for several months during the early stages of a project; for successful products, downloads eventually sharply increase (e.g., Golder and Tellis 1997; Agarwal and Bayus 2002). In most cases, the takeoff in downloads can be visually identified\(^7\). For example, the takeoff in downloads for *MegaZeux* occurs at 16 months and *RadeonTweaker* has a takeoff at 6 months. All the open source products in our sample either exhibited a similar takeoff in downloads (649 projects), or had zero downloads for the entire observed period (336 projects). For the projects with a takeoff in downloads, the mean (median) time to takeoff is 13 (9) months.

To explore the relationship between *user eyeballs* and development activity, we compare our development activity measures before (when there are few user eyeballs) and after\(^8\) (when there are many user eyeballs) the download takeoff. Thus, for this analysis we only consider open source projects that exhibited a takeoff in downloads. A

---

\(^7\) As discussed in Agarwal and Bayus (2002), we follow the procedure outlined in Gort and Klepper (1984) to identify the download takeoff point for each product in our sample. Basically, this procedure is a systematic search for the first occurrence of a very large increase in downloads after a product is made available.

\(^8\) Our measures of development activity are based on project information for twelve months after the download takeoff. Constructing our development activity measures using all available information (up to December 2003) gives the same conclusions as those reported here.
cross-tabulation of development activity before and after takeoff is in Table 4. Based on a statistical test of two proportions\(^9\), it is significantly more likely that there are bug reports, bug fixes, support request, commits, and version releases \textit{before} takeoff than after takeoff. In other words, product development activity is more prevalent before there is a large number of downloads, i.e., when there are few user eyeballs. Interestingly, this result is consistent with the product development literature (e.g., Klepper 1997; Krishnan and Ulrich 2001; Agarwal and Bayus 2002; Wynn 2004), but is contrary to some of the bazaar development principles as more user eyeballs fails to translate into more bug reports, bug fixes, or support requests (e.g., Raymond 1998).

To examine the robustness of our finding that “user eyeballs don’t matter,” we examine development activity for different levels of development effort before and after the download takeoff in Table 5. Generally speaking, open source projects with more than developer are significantly more likely to have development activity before the download takeoff than after the takeoff (in Table 5, compare the before and after takeoff columns for projects with more than one developer). In addition, Table 5 confirms our results in Section in 3.2 that developer eyeballs are significantly related to development activity \textit{before the takeoff in downloads} (but \textit{not} after the download takeoff). This can be seen by statistically comparing the proportions within each column (e.g., 20.7\% of the projects with one developer and one project manager had at least one bug report before the takeoff, whereas 48.1\% of the projects with one project manager and more than one developer had at least one bug report before the takeoff).

\(^9\) A paired-sample t-test gives the same results.
In summary, we report three important and related findings: (1) strong empirical evidence that user eyeballs are not significantly associated with product development activity, (2) developer eyeballs are positively related to development activity only when there are few user eyeballs and (3) development activities significantly drop off once downloads dramatically increase and that project development after the download takeoff does not really depend on development effort.

One possible explanation for our first finding indicating a lack of impact of user eyeballs on development activity is that today the profile of a typical user is very different from that of early open source projects (e.g., Linux during the early 90s). Users today can be described as being passive eyeballs. This is in contrast to the hacker community members that served as not only as developers but also as primary users for the early projects, and who were technologically adept and could potentially play an active role as users. But, with the adoption of open source products well beyond the hacker group, typical users are now less sophisticated in the workings of the technology (relative to the hacker group) and hence, are more interested in using the software for their needs rather than participating as beta testers. Even commercial vendors that use beta-testers usually rely on the lead user group and not their typical user. One implication of these results is that studies simulating the open source development process should be cautious in their assumptions regarding the impact of a large user population (e.g., Dicker, 2004)

Our second finding of a positive relationship between developer eyeballs and development activity only in the case of low user eyeballs gives us a more refined understanding of the importance of developer eyeballs. Our third result provides an
initial answer to the question raised by Garcia (2004) about whether product acceptance leads to further increases in project community size and product development. Our results do not support this idea. Instead, we find that development activity after a takeoff in downloads is substantially lower than the activity before takeoff.

3.4 Development Activity and Product Adoption

We now explore the relationship between development activity and product adoption. In line with Wynn (2004) and Garcia (2004), we use cumulative downloads as our measure of product adoption (recognizing the limitations as discussed in the previous section). It is important to point out that our data are truncated (i.e., right-censored) in that we stopped observing download activity in December 2003, i.e., for numerous products in our sample we do not observe when (or whether) a download takeoff occurs at some later time. Consequently, we need to employ an appropriate statistical methodology to account for this kind of potential sample selection bias\(^{10}\).

We follow the general analysis approach used by Golder and Tellis (1997) and Agarwal and Bayus (2002) to study the timing of a sales takeoff for product innovations. Cox’s (1972) proportional hazards regression model is used to study the timing of a take-off in downloads (i.e., speed of product adoption). The proportional hazards model is appropriate since it allows for estimation of the determinants of the hazard rate, i.e., the conditional probability of take-off in month \(t\) given that the product has not taken off till month \(t-1\).

For the \(i^{th}\) open source product, the hazard rate function \(h_i(t)\) is defined as

\[
\log h_i(t) = \log h(t; x_i) = \alpha(t) + x_i' \beta
\]  

\(^{10}\) Using the hazard model framework discussed in this section, we estimate that the average (median) time to a download takeoff for our entire sample of open source projects is 16.7 (13) months.
where $\alpha(t)$ is an arbitrary and unspecified baseline hazard function, $x_i$ is a vector of measured explanatory variables for the $i$th product, and $\beta$ is the vector of unknown coefficients to be estimated. The risk ratio, calculated as $e^\beta$, gives the marginal effect of each explanatory variable on the hazard rate, i.e., a risk ratio greater than 1 ($\beta>0$) indicates the percentage increase in the hazard rate due to a unit increase of the corresponding explanatory variable, while a risk ratio less than 1 ($\beta<0$) indicates the percentage decrease in the hazard rate, respectively. Following Golder and Tellis (1997) and Agarwal and Bayus (2002), we do not include a term for unobserved heterogeneity since we only analyze non-repeated events. Parameter estimation is accomplished using the partial likelihood method as implemented in the SPSS package.

Because we want to specifically test Raymond’s (1998) development bazaar principle of the importance to “release often and release early,” we consider the average rate of version releases before download takeoff and the time to first version release before takeoff as explanatory variables in the hazard model. We also examine the role of bug reports, bug fixes, support requests, and commits before download takeoff in explaining the speed of open source product adoption.

Estimation results for single variable models, as well as all the explanatory variables together, are reported in Table 6. With the exception of bug reports and bug fixes, all the other measures of development activity are significantly related to the

---

11 Note that the risk ratio is interpreted as a deviation from mean values, i.e., the effect on the hazard rate as a change from the average hazard rate function.

12 For each project, this variable is calculated as the number of version releases before takeoff divided by the number of months to takeoff.
conditional probability of download takeoff. In agreement with Raymond's (1998) dictum, open source projects with a high rate of version releases before takeoff have quicker download takeoff times (the estimated coefficient for the rate of version releases in Table 6 is positive) and projects with earlier releases have faster takeoff times (the estimated coefficient for the time of first version release in Table 6 is negative). Interestingly, we also find that projects with support requests and outstanding commits have relatively slow download takeoffs. This suggests that downloads tend to be delayed for open source projects which are still being improved.

4. Conclusions and Directions for Future Research

In sum, our study finds support for key elements of the bazaar model of open source development. The implications for open source project managers is that attracting a larger number of developers is important in achieving higher levels of product development activity. Also, project developers may note that product success in terms of large number of users (downloads) can be achieved by higher levels of development activity. But, product success is also associated with a drop off in development activity, suggesting that further product development and improvement is not the norm. It seems that typical open source products reach a level of being “just good enough,” and developers then turn their attention to new projects which are more interesting and challenging. Contrary to the tenets of the bazaar model, we find that “user eyeballs” do not significantly contribute to increased development activity. Thus, our results also suggest that the bazaar community development model involving
developers and users originally proposed by Raymond needs revision for the more
typical open source software project.

Although our study indicates that developer eyeballs are positively related to
development activity and that project activity is related to product adoption, research
exploring other measures of project community size (developer and user eyeballs),
development activity, and product adoption should be conducted to confirm our findings.
Future research might also extend our work by examining various measures of project
complexity and how it may influence these relationships (e.g., see Jorgensen 2001).

As the developer community size increases so do the number of peripheral\textsuperscript{13} members. In particular, future research might explore the idea that above a threshold
the number of additional developer eyeballs may have diminishing returns in terms of
development activity (owing to the disproportionate increase in the number of peripheral
members).

The underlying reason for why development activity after a takeoff in downloads
is substantially lower than the activity before takeoff is an important question that can be
explored. One way this can be approached is to examine if the motivation levels of the
participants has changed after there is a sharp increase in download activity.

Our results also indicate that many small (e.g., one-person) teams of open
source projects do achieve product success. This raises important questions as to how
and why some of these projects succeed with apparently minimal resources. There is
some evidence pointing to knowledge “reuse” across projects that benefits the smaller
teams as they take advantage of the work of larger communities and are able to
leverage the code already generated (e.g., Brown and Booch 2002). So, while the

\textsuperscript{13} Individuals that contribute on less than a regular basis to the project.
apparent community size may seem to be small, the actual development participation (including indirect knowledge and experience gained through other projects) might really be higher. This aspect of small team success needs to be further explored.

Another interesting aspect in this domain is that a myriad of open source products have been initiated as an outgrowth of other successful projects. For example, the popular open source database MySql has spurred facilitating products like PhpMyAdmin that supports the administrative functions of the database engine. Shocker, Bayus, and Kim (2004) suggest that facilitating products can positively influence the success of existing products. Future research might study such intra-product relationships and whether open source products that have the support of facilitating products experience quicker download takeoff.

Finally, our study suggests some directions for researchers interested modeling the market evolution of open source products. A look at the cumulative download patterns in Figure 2 suggests that some products may conform to the widely used S-shaped diffusion curve (MegaZeux), while others may not follow the standard diffusion model (e.g., RadeonTweaker and LibSIMD). Future research might explore the applicability of different new product diffusion models to open source innovations.
References


**Figure 1**
Research Model

- **Project Community Size**
  - Developer eyeballs
  - User eyeballs

- **Development Activity**

- **Product Adoption**
Figure 2
Examples of the Download Takeoff for Some Open Source Products
### Table 1
Number of Developers and Project Managers

<table>
<thead>
<tr>
<th>Number of Project Managers</th>
<th>Number of Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td></td>
<td>51.4% (n=505)</td>
</tr>
<tr>
<td></td>
<td>22.2% (n=218)</td>
</tr>
<tr>
<td>More than One</td>
<td>0.0% (n=0)</td>
</tr>
<tr>
<td></td>
<td>26.4% (n=260)</td>
</tr>
</tbody>
</table>

### Table 2
Project Activity

<table>
<thead>
<tr>
<th>Project Activity</th>
<th>0</th>
<th>≥1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Bug Fixes</td>
<td>61.8% (n=446)</td>
<td>38.2% (n=276)</td>
</tr>
<tr>
<td>Number of Bug Reports</td>
<td>52.0% (n=386)</td>
<td>48.0% (n=357)</td>
</tr>
<tr>
<td>Number of Support Requests</td>
<td>82.9% (n=561)</td>
<td>17.1% (n=116)</td>
</tr>
<tr>
<td>Number of Commits</td>
<td>36.5% (n=292)</td>
<td>63.5% (n=509)</td>
</tr>
<tr>
<td>Number of Versions Released</td>
<td>17.8% (n=156)</td>
<td>82.2% (n=720)</td>
</tr>
</tbody>
</table>
Table 3
Cross-tabulations of Project Activity and “Developer Eyeballs”

<table>
<thead>
<tr>
<th></th>
<th>1 Developer</th>
<th>&gt;1 Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>At least 1 bug fix</td>
<td>21.0%* (n=328)</td>
<td>41.9% (n=322)</td>
</tr>
<tr>
<td>At least 1 bug report</td>
<td>29.1% (n=327)</td>
<td>52.2% (n=322)</td>
</tr>
<tr>
<td>At least 1 support request</td>
<td>10.1% (n=328)</td>
<td>17.7% (n=322)</td>
</tr>
<tr>
<td>At least 1 commit</td>
<td>36.6% (n=328)</td>
<td>71.7% (n=322)</td>
</tr>
<tr>
<td>At least 1 version release</td>
<td>69.7% (n=505)</td>
<td>77.2% (n=478)</td>
</tr>
</tbody>
</table>

*21.0% of projects with 1 developer had at least 1 bug fix, and 100 - 21.0 = 79% had no bug fixes.
### Table 4
Cross-tabulations of Project Activity and “User Eyeballs”
(only projects that exhibited a takeoff in downloads)

<table>
<thead>
<tr>
<th></th>
<th>Before Download Takeoff</th>
<th>After Download Takeoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>At least 1 bug fix</td>
<td>27.1%* (n=649)</td>
<td>12.5% (n=431)</td>
</tr>
<tr>
<td>At least 1 bug report</td>
<td>35.6% (n=648)</td>
<td>17.2% (n=431)</td>
</tr>
<tr>
<td>At least 1 support request</td>
<td>10.3% (n=649)</td>
<td>6.3% (n=431)</td>
</tr>
<tr>
<td>At least 1 commit</td>
<td>52.5% (n=649)</td>
<td>20.9% (n=431)</td>
</tr>
<tr>
<td>At least 1 version release</td>
<td>92.6% (n=649)</td>
<td>29.4% (n=431)</td>
</tr>
</tbody>
</table>

*27.1% of projects had at least 1 bug fix before the download takeoff, and 100-27.1=73.9% had no bug fixes before the download takeoff.
Table 5  
Cross-tabulations of Project Activity, “Developer Eyeballs,” and “User Eyeballs”  
(only projects that exhibited a takeoff in downloads)

<table>
<thead>
<tr>
<th></th>
<th>Before Download Takeoff</th>
<th>After Download Takeoff</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>At least 1 bug fix</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Project Mgr &amp; 1 Developer</td>
<td>12.9%* (n=295)</td>
<td>14.7% (n=191)</td>
</tr>
<tr>
<td>1 Project Mgr. &amp; &gt;1 Developer</td>
<td>38.3% (n=162)</td>
<td>13.0% (n=108)</td>
</tr>
<tr>
<td>&gt;1 Project Mgr. &amp; &gt;1 Developer</td>
<td>39.5% (n=192)</td>
<td>9.1% (n=132)</td>
</tr>
<tr>
<td><strong>At least 1 bug report</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Project Mgr &amp; 1 Developer</td>
<td>20.7% (n=294)</td>
<td>18.8% (n=191)</td>
</tr>
<tr>
<td>1 Project Mgr. &amp; &gt;1 Developer</td>
<td>48.1% (n=162)</td>
<td>15.7% (n=108)</td>
</tr>
<tr>
<td>&gt;1 Project Mgr. &amp; &gt;1 Developer</td>
<td>47.9% (n=192)</td>
<td>15.9% (n=132)</td>
</tr>
<tr>
<td><strong>At least 1 support request</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Project Mgr &amp; 1 Developer</td>
<td>4.4% (n=295)</td>
<td>7.3% (n=191)</td>
</tr>
<tr>
<td>1 Project Mgr. &amp; &gt;1 Developer</td>
<td>13.0% (n=162)</td>
<td>6.5% (n=108)</td>
</tr>
<tr>
<td>&gt;1 Project Mgr. &amp; &gt;1 Developer</td>
<td>17.2% (n=192)</td>
<td>4.5% (n=132)</td>
</tr>
<tr>
<td><strong>At least 1 commit</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Project Mgr &amp; 1 Developer</td>
<td>29.8% (n=295)</td>
<td>18.3% (n=191)</td>
</tr>
<tr>
<td>1 Project Mgr. &amp; &gt;1 Developer</td>
<td>70.4% (n=162)</td>
<td>29.6% (n=108)</td>
</tr>
<tr>
<td>&gt;1 Project Mgr. &amp; &gt;1 Developer</td>
<td>72.4% (n=192)</td>
<td>17.4% (n=132)</td>
</tr>
<tr>
<td><strong>At least 1 version release</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Project Mgr &amp; 1 Developer</td>
<td>94.9% (n=295)</td>
<td>27.2% (n=191)</td>
</tr>
<tr>
<td>1 Project Mgr. &amp; &gt;1 Developer</td>
<td>87.7% (n=162)</td>
<td>29.6% (n=108)</td>
</tr>
<tr>
<td>&gt;1 Project Mgr. &amp; &gt;1 Developer</td>
<td>93.2% (n=192)</td>
<td>32.6% (n=132)</td>
</tr>
</tbody>
</table>

*Of the projects with 1 project manager and 1 developer, 12.9% had at least 1 bug fix before the download takeoff, and 100-12.9=77.1% had no bug fixes before the download takeoff.
Table 6
Cox Proportional Hazard Analysis of Product Adoption Timing
(standard errors in parentheses)

<table>
<thead>
<tr>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
<th>Model 4</th>
<th>Model 5</th>
<th>Model 6</th>
<th>Model 7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bug Fixes Before Takeoff</strong> (binary: 0 or ≥1)</td>
<td>-0.16 (0.09)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-0.32 (0.16)</td>
</tr>
<tr>
<td><strong>Bug Reports Before Takeoff</strong> (binary: 0 or ≥1)</td>
<td>-</td>
<td>-0.06 (0.08)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-0.11 (0.15)</td>
</tr>
<tr>
<td><strong>Support Requests Before Takeoff</strong> (binary: 0 or ≥1)</td>
<td>-</td>
<td>-</td>
<td>-0.36a (0.13)</td>
<td>-</td>
<td>-</td>
<td>-0.42a (0.14)</td>
</tr>
<tr>
<td><strong>Commits Before Takeoff</strong> (binary: 0 or ≥1)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-0.16 (0.08)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Average Rate of Version Releases Before Takeoff</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.34a (0.05)</td>
<td>-</td>
</tr>
<tr>
<td><strong>Time to First Version Release Before Takeoff</strong> (months)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-0.04a (0.01)</td>
</tr>
<tr>
<td><strong>-2 LL χ²</strong></td>
<td>7608.0 3.27 (p=0.07)</td>
<td>7597.2 0.43 (p=0.51)</td>
<td>7603.3 7.47 (p&lt;0.01)</td>
<td>7607.3 4.04 (p=0.05)</td>
<td>7147.6 44.86 (p&lt;0.01)</td>
<td>7059.1 26.79 (p&lt;0.01)</td>
</tr>
</tbody>
</table>

a significant at 0.01 level or better