2005

Are All Open Source Projects Created Equal? Understanding the Sustainability of Open Source Software Development Model

Ju Long
Texas State University - San Marcos, julong@txstate.edu

Michael Juntao Yuan
JBoss, michael.yuan@enterprisej2me.com

Are All Open Source Projects Created Equal? Understanding the Sustainability of Open Source Software Development Model

Ju Long
Texas State University-San Marcos
julong@txstate.edu

Michael Juntao Yuan
JBoss
Michael.Yuan@enterprisej2me.com

ABSTRACT

A very intriguing question in Open Source software (OSS) development is: why there are only a few open source projects succeed, while the majority of projects never do. In this research, we examine the factors that may influence the performance of OSS projects. We particularly focus on the OSS's core developers' role in the project's success. Extant research has yet to distinguish core developers and non-core developers from the community at large. The different roles of the core developers and non-core developers in OSS projects' success still remain unclear. Our research contributes to the literature by separating the core developers from the development forces in general and empirically examining the core developers' importance. Drawing the evidences from our extensive dataset of 300 open source projects, we demonstrated that core developers' leadership and project advocation are crucial in determining the fate of the OSS projects. Our research could provide better understanding of OSS sustainability. It could also give practical advice to the OSS community on how to make the project successful.

Keywords

Software Development, Open Source Software, Sustainability, Development Model

1. INTRODUCTION

1.1. Open Source: A Disruptive Paradigm

Compared with traditional proprietary software development model, the open source software (OSS) model is a radically new paradigm to develop software (Raymond, 1997; Moody, 2001; Sharma, et al. 2002). In the OSS development process, software source code is freely available for anyone to view, download, modify and re-distribute as long as it is under the same open source license (http://www.opensource.org). Most open-source software projects rely entirely on the voluntary efforts of a community of developers, although some projects are coordinated and led by commercial entities. Such a voluntary community process keeps the cost of development and testing low. The nearly zero total cost of ownership (TCO) gives open source software a strong edge in the competition in the software industry.

1.2. Romanticized picture of Open Source software development

A few projects initiated in open source community, such as GNU, Linux, Apache, MySQL and PHP, have achieved extraordinary success and are among the most prominent software used in the technology industry. For example: Apache, a powerful server side software, runs more than 60 percent of all websites in the world.

However, except for a few truly spectacular successful projects, the majority of the open source projects are lackluster, with no active developing activity at all. Many die at the beginning, while others survive, but with little momentum behind them (Thomas and Hunt, 2004, Healy and Schussman, 2003).

1.3. Research motivation

Why some of the open source projects could achieve success while most of the other open source projects cannot? What are the factors that could influence the success or failure of the open source projects? Several research has tried to address these questions before and has generated very insightful results (e.g. Crowston et al., 2003; Hann et al., 2002; Stewart et al., 2005). Our research is different from these prior literature in two important ways:

First, we differentiate the OSS development force into core developers and non-core developers from the community. We argue the importance of this differentiation because of these two developer groups' very distinctive roles in the OSS development process. We also set to empirically prove that the core developers have a very crucial role in deciding the project's success. Prior research on open source software development has not addressed the distinction between core and non-core developers, nor have they empirically examined the core developers' impact on OSS project's fate.

Second, our research is based on a dataset of 300 open source projects. These projects not only include the successful projects but also those less successful ones. To our knowledge, very few prior research has examined the projects from both realms side by side.

The rest of the paper are organized as follows: in section 2, we review the existing theories on open source software development, especially those based on organizational theories, to decide potential factors that could influence the success of the projects. In section 3, we discuss the empirical study and data analysis results. In section 4, we conclude our paper and examine the future research topics.

2. HYPOTHESES DEVELOPMENT

Compared with traditional software development model, open source model has very distinctive organizational structure, development process and culture. In the traditional proprietary software development, there is only one development entity – the software developers, while in open source development, there are two distinctive development groups: a small number of core developers (usually less than 15 people) and a large number of anonymous and volunteer developers from the community at large. In order to identify the factors that may influence the success or failure of an open source software project, it is important to distinguish these two main development entities from each other. We study the two entities and develop hypotheses according to the common organizational dimensions (March and Simon, 1958; Mintzberg, 1971; Nohria, 1995, Srinarayan, et al. 2002), including division of labor, co-ordination mechanisms, distribution of decision-making authority, organizational boundary and development process.

Open source projects are drastically different from traditional software development projects mainly because they rely on a large number of anonymous developers from the community to make voluntary effort in developing the projects. These developers are organized into a very loosely centralized and networked community - a "Bazaarr" (Raymond, 1997, Tirole and Lerner, 2002). There is no formal development plan or schedule to follow strictly (Mockus et al. 2000; Schmidt and Porter, 2001). The developers choose the projects they want to participate, decide their effort level according to their own schedule. This open organizational structure encourages new contributors to participate in the projects. By remaining open to new contributors, the project could have an unlimited supply of innovative ideas (Fielding, 1999; Raymond, 2001).

Several activities could indicate how active the non-core developers are.

First is how active the developer forums are. In the open source software development process, there is not a central decision maker. Developers in the community make judgment on what tasks to do and how to do it (Fielding, 1999; Markus et al., 2000; Mockus et al., 2000). To communicate with each other, developers in the community use online forums as the essential channels to coordinate development activities. Many development tasks are assigned through the forum. Many development decisions are made through the forum. To maintain an active online forums are crucial to keep the communication channels open for the developers at large. Thus the number of messages posted in the forum could be used to calibrate how active the developers from the community are. Hence, our first proposition is:

H1a: the number of messages in the forum is positively correlated with the OSS project's success.

Reporting bugs and patches are the two main ways the non-core developers contribute to the OSS process. A large number of bug reports and patch report generally shows that the project attracts a lot of attention from the developer community.

Therefore, we propose that:

H1b: the number of bug report from the non-core developers is positively related to OSS project's success.

H1c: the number of patch report from the non-core developers is positively related to OSS project's success.

All of these three hypotheses are related to the non-core developers' contribution to OSS project's success.

However, because of non-core developers' unique role in the OSS development, attracting talented developers and keep them constantly motivated in participating in the project becomes a crucial factor in the project's success. This is especially important in the early stages of the development before the number of the project's developers could reach the critical mass. Core developers are the ones who attract and motivate developers in the projects.

One important way is to update the projects and release a newer version of the project frequently. The more frequent updates show more effort from the core developers to promote the project and motivate developers in the community. Frequently updated projects often attract more developers to participate in the projects. This is simply because developers see the project is very active so they know their efforts could be valued. New version of files also means new challenges and tasks for the developers to work on.

Like frequently releasing new files, broadcasting news is also an important way for the core developers to keep the project active and motivate developers to participate in the project.

Thus, we propose the following hypotheses with regard to the core developers' efforts in promoting and motivating developers:

H2a: The number of file releases is positively related to OSS project's success.

H2b: The number of news release is positively related to OSS project's success.

To test the effects of core developers' activeness on the project development, we focus on two areas:

In the open source software development process, there is not a central decision maker. Core developers make judgment on what tasks to do and how to do it (Fielding, 1999; Markus et al., 2000; Mockus et al., 2000). To communicate with each other, developers use e-mail lists as the essential channels to coordinate development activities. Maintaining the email list are core developers' communication channels. It is a good indicator of the core developer's active level. Therefore,

H3a: The number of email list is positively related to OSS project's success.

Another important tool in coordinating the core developers' development efforts is the concurrent version system (CVS). The total number of CVS updates accurately indicates how active the core developers are. Thus,

H3b: The number of CVS is positively related to OSS project's success.

An important feature of open source software development projects is their self-governance (Markus et al., 2000; Cook, 2001; Raymond, 2001). Compared with the open source projects, most corporate projects have much stronger belief in central planning. Open source projects are different. They almost always start with a single person at their center. If, after the first couple of releases, they start to grow, people might volunteer to join (Thomas and Hunt, 2004).

However, that does not mean that there is no centralized decision making. The core developers set the goal and mission of the projects. The right mission attracts more developers to join in. Core developers also are responsible for dividing the projects into different tasks to be handled by individuals or teams. Coordination among these tasks is also the responsibility of core developers. All these activities cannot be handled by one or two core developers. The more core developers participating means more talents and resources to fulfill these responsibilities.

Therefore, we posit that:

H4: The number of active core developers should have positive impact on OSS project's success.

Besides the measurements with regard to core and non-core developers, we also think that two characteristics of the project itself may influence its success.

One is development status: by Sourceforge's criteria, project's development status is divided into 7 stages: planning, pre-alpha, alpha, beta, production/stable, mature and inactive. We excluded those projects in the inactive stage because we only compare the active projects' current activities and historical data. We posit that the more mature the project is, the more possible it will be success. Thus,

H5: The development status of the project is positively related to the project's success.

The other is the project life span, i.e. how long the project has been created. The longer the projects has been active, the more possible it has established its development base and been successful. Therefore,

H6: the project's life span is positively related to the project's success.

3. DATA COLLECTION, MODEL BUILDING AND EMPIRICAL ANALYSIS

3.1. Data Collection Process

3.1.1. SourceForge.Net as the data collection site

Empirical data is collected from SourceForge.net website. SourceForge.Net is the world's largest open source software development project host site, with the largest repository of open source projects. SourceForge.net provides a centralized place for open source developers to control and manage their projects. There are a total number of 89,103 open source projects hosted in SourceForge.net with more than 935,651 registered developers (as of 10/17/2004 data). It thus provides us the best research site to collect empirical data on various open source projects' performance and attributes.

3.1.2. Sampling Procedure

Our data sample consists of 300 open source software development projects hosted in the Sourceforge.Net. They are the first 300 active projects ranked by Sourceforge.Net. These 300 projects provide a rich data set that includes the most successful open source projects as well as many less prominent projects. However, unlike the other projects that are ranked lower than 300, these projects are active enough to provide usable data on the project activities. If we randomly sampled all the open source projects, the data would have been barely usable because the majority of the open source projects do not have any activity at all (as we have discussed in the introduction section). Values for important variables such as number of download and number of bug report are almost near 0. By focusing our sample on the first 300 projects, we could ensure our sample's usability as well as its validity.

3.2. List of the Variables

SourceForge provides very detailed information on each project's activities across the project's entire time span. We collect all the major variables discussed in the hypotheses building section. Our dependent variable in the model to measure the success of the projects is the number of downloads. It is an essential variable to show how successful the project is. Generally, more number of downloads means a more successful project.

Independent variables include: development status, project lifespan, number of developers, number of messages in the forums, number of mailing list, number of bug report, number of patch report, number of CVS report, number of file releases and also number of news release.

3.3. Empirical Analysis

3.3.1. Data Transformation

To check the data quality and validate the assumptions for the statistical procedures, we use the frequencies procedure to obtain the summary of each individual variable.

As shown on the frequency table before data transformation (Table 1.1) and Figure 1.1, the mean is quite different from the median for every variable, suggesting the distribution of the variable value is asymmetric. The large positive skewness confirms that. This is because the data set includes some of the most popular projects, whose activity data are skewed toward the left hand side.

	Forum	Download	Bug	Patch	CVS	Release	News
N Valid	151	291	299	299	293	273	220
N Missing	149	9	1	1	7	27	80
Mean	577.54	355017.53	212.97	34.02	4071.81	18.34	10.39
Median	114	24545	49	3	1413	12	6
S.D.	1378.12	1839506.96	650.62	189.38	11014.05	21.03	14.77
Skewness	4.39	10.16	7.34	12.12	8.32	2.55	4.24

Table 1.1: Frequency table before data transformation

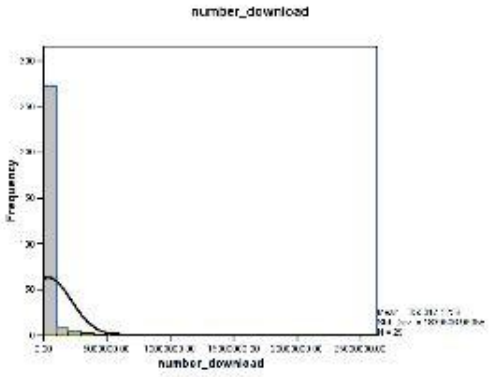

Figure 1.1: Histogram of number of download before log-transformation

The large positive skewness could inflate the standard deviation to a point where it is no longer useful as a measure of the spread of data values. In order to increase the reliability of the data analysis, we conduct a transformation so that we can bring the distribution of the variable values closer to normal.

The log transformation is a sensible choice because the variables take only positive values and are right skewed. Table 1.2 and Figure 1.2. show the frequency table of the variables and histogram after the log transformation. We can see that the transformation has brought the distribution closer to normal.

	Forum	Download	Bug	Patch	CVS	Release	News
N Valid	151	287	290	197	288	272	218
N Missing	149	13	10	103	12	28	82
Mean	4.79	10.32	4	2.28	7.2	2.37	1.82
Median	4.74	10.16	4	2.2	7.29	2.48	1.79
S.D.	1.81	2.12	1.63	1.63	1.54	1.1	1
Skewness	0.21	0.23	0.92	0.51	-0.27	-0.26	0.22

Table 1.2: Frequency table after data transformation.

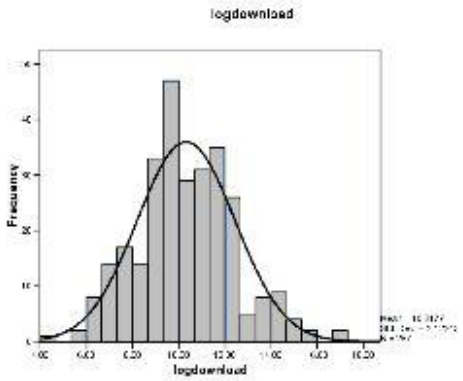

Figure 1.2: Histogram of number of download after log-transformation

3.3.2. Data reduction – factor analysis

In our hypotheses development, we set to predict the performance of the projects based on a set of predictors. However, many of these variables are correlated. We use factor analysis to remove redundant and highly correlated variables from the data file, and replace the data file with a smaller number of uncorrelated variables. During the factor analysis process, we will also be able to examine the latent variables that are underlying the relationships between the manifest variables.

The variables we put into factor analysis include the following ones: development status, number of developers, project life-span, number of messages posted on the forum (log transformed), number of bug report (log transformed), number of patch report (log transformed), number of CVS (log transformed), number of file release (log transformed), number of news release (log transformed) and number of mailing lists,

Judging from the Extraction communalities table (Table 1.3), we extract five components (four of them with eigen-value larger than 1, and one with eigen-value as .933) and they account for a total of 10 variables. These five components explain nearly 78% of the variability in the original 10 variables. We also use scree plot (Figure 1.3) to confirm the optimal number of components.

Component	Initial Eigenvalues			Rotation Sums of Squared Loadings		
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %
1	3.633	36.330	36.330	2.205	22.047	22.047
2	1.245	12.452	48.781	2.143	21.432	43.479
3	1.113	11.126	59.908	1.262	12.624	56.104
4	.933	9.327	69.235	1.122	11.220	67.324
5	.849	8.493	77.728	1.040	10.404	77.728
6	.700	7.003	84.731			
7	.557	5.572	90.303			
8	.387	3.866	94.169			
9	.359	3.590	97.759			
10	.224	2.241	100.000			

Table 1.3: Extraction communalities table

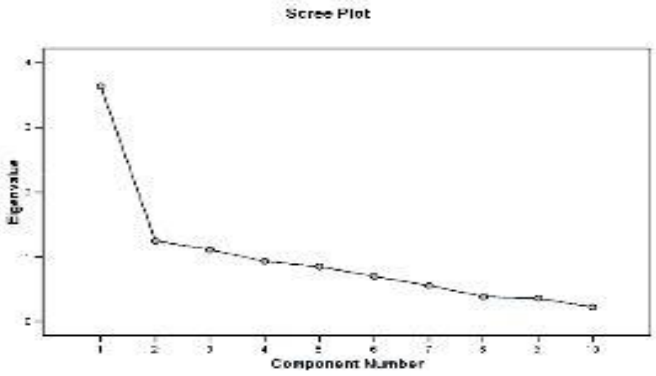

Figure 1.3: Scree plot

After extracting the components, the rotated component matrix helps us to determine what the components represent.

The first component is most highly correlated with a number of forum and number of patch report, and also number of bug report. All these three components are associated with the activities from the non-core developers in the open source community.

The second component is associated with number of developers, number of CVS update, as well as number of mailing list. These three variables show the strength of the core developers of the projects.

The third component is correlated with the number of file release and number of news release. These two variables mainly describe the core developers' activities in promoting and publicizing the project.

The fourth component is associated with one variable: development status. The fifth component is also associated with one variable only: project life span.

The relationship between these extracted components and their variables are demonstrated in Figure 1.4. Note that in the square are the manifest variables and the latent (extracted components) are in the circle.

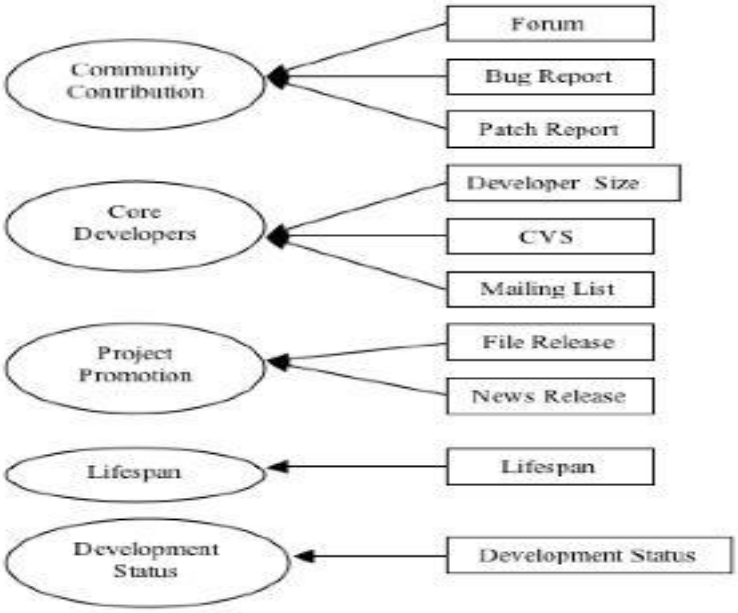

Figure 1.4: Latent variables and manifest variables

We then use the component score variables in places of the original variables to avoid the linearity in the regression analysis. We check the plots of the component scores for outliers and non-linear associations between the components. Judging from the scatter plot matrix of the component scores (Figure 1.5), we did not see abnormalities in the component scores.

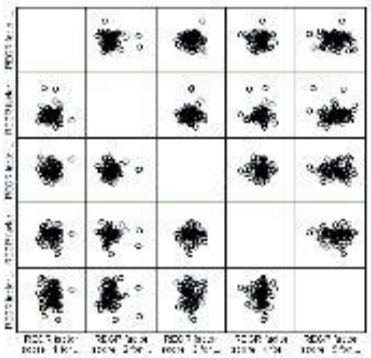

Figure 1.5: Scatter plot matrix of the component scores

3.3.3. Regression analysis

Linear regression is used to model the value of the dependent variable – success of the projects – with its linear relationship to one or more predictors. We assume that there is a linear relationship between the dependent variable and each predictor.

As we specified before, we use number of download as the indicator of how successful the project is. Thus, log-transformed number of downloads is the dependent variable in the model. The independent variables, i.e. the predictors are the factors we extracted in the factor analysis. We will use the component score of each factor in the model.

The ANOVA tables (Table 1.4 and Table 1.5) reports a significant F statistic, indicating that a strong prediction power of the predictors. As a whole, the regression does a good job of modeling performance of the project. Nearly half of the variation (R Square=.465) in download times is explained by the model.

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.682(a)	.465	.423	1.53624

Table 1.4: ANOVA

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	131.077	5	26.215	11.108	.000(a)
	Residual	151.042	64	2.360		
	Total	282.119	69			

Table 1.5: ANOVA 2

Judging from the coefficient table (Table 1.6), we can determine whether the predictors are significant ones. As we proposed, the contribution from the community, the core developers' activities as well as the project promotion all play significant roles in generating more download and make the project a success. Thus the results support hypotheses H1 through H4. Project's lifespan is less significant. The significant value is .169. Project's development status is quite significant. This is easy to explain since the more advanced projects usually could be able to attract more download.

Model		Unstandardized Coefficients	Standardized Coefficients		T	Sig.
		B	Std. Error	Beta		
1	(Constant)	11.071	.184		60.295	.000
	Community Contribution	.828	.185	.410	4.480	.000
	Core Developers	.745	.185	.368	4.027	.000
	Project Promotion	.620	.185	.306	3.350	.001
	Project Lifespan	.257	.185	.127	1.391	.169
	Development Status	.457	.185	.226	2.469	.016

Table 1.6: Coefficient of ANOVA

Checking the multicollinearity test (see Table 1.7), we can see there is not significant multicollinearity.

Model		Correlations			Collinearity Statistics	
		Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)					
	Community Contribution	.410	.489	.410	1.000	1.000
	Core Developers	.368	.450	.368	1.000	1.000
	Project Promotion	.306	.386	.306	1.000	1.000
	Project Lifespan	.127	.171	.127	1.000	1.000
	Development Status	.226	.295	.226	1.000	1.000

Table 1.7: Collinearity test

4. DISCUSSION OF THE RESULTS AND CONCLUSION

4.1. Importance of the developers' contribution from the community

Our analysis confirmed the importance of the developers' contribution from the community in the success of the open source projects. This result is consistent with the unique organizational structure and culture of open source development model. Unlike proprietary software development model, the essence of open source model is the thousands of developers in the community and their voluntarily contribution to the projects. Our research not only confirms the importance of the developers from the community; we also quantitatively measure the importance.

Furthermore, we identify several manifest factors that indicate the activeness of the developers contributions, including patch report, bug report as well as forum activities:

H1a is confirmed. The number of messages in the forum is positively correlated with the OSS project's success. We can further learn that as core developers are the main facilitator of the public forum, it is important for them to make an effort to maintain the active forum, to keep the non-core developers' communication channel open.

H1b and H1c are both confirmed too, which shows us that numbers of patch and bug reports could be reliable predictors of the success of the projects.

4.2. Importance of the core developers

More important in our findings is that we stressed the importance of the core developers in the projects. In existing research, core developers' role is often under-explored. It is common to believe that anonymous developers are the essential part of open source project development, and the core developers play less important or even marginal roles. However, our analysis demonstrates the crucial role that the core developers play. It is the second most important factor in deciding whether a project could success or not.

We also show several factors are very significant indicators in determining the success of the projects on the core developers side. As we confirmed in H3a and H3b, the core developers' activeness showed by their email activities and CVS report are significant indicators of the project's performance. As we confirmed H4, the mere number of core developers is also a significant indicator.

4.3. Crucial role of promoting the projects

Another important finding of our research is the importance of core developers' actively promoting and publicizing open source projects. In the open source development community, thousands of projects are competing for developers' attention

and contribution. For a project to success, the core developers need to actively promoting their project in the community. It is important for the projects to send signals to the developers at large that the project is active and developing fast. As we identified and confirmed in H2a and H2b, there are two important signals that the core developers could send to the community: the frequent release of new files and frequent release of the news about the projects.

4.4. Contributions of this research

This research systematically examine why certain open source projects succeed while the majority of the projects fail. Our research identified several factors that influence the performance of the projects, including the less-explored role of core developers and the importance of promoting the projects. The contribution of this research include:

First, our research could shed light to the academic research in the open source field. Our research is one of the first research to distinguish core developers from the non-core developers in the community at large, and study these two developer groups' very distinctive roles in the OSS project success. Moreover, we compare the successful projects with the less successful ones and identify a series of factors that could influence the performance of the projects, which is also unique.

Second, for industry researchers and practitioners, our research could be useful in predicting which projects have more potential to succeed, and consequently decide which projects to invest or support.

Third, open source developers could learn from our research the factors deciding the fate of their projects. For example, developers could learn from our study the crucial role of promoting and publicizing the projects in the community, and release the project files more often. They could also try to set up a more efficient reputation system to motivate the developers from the community to participate in the projects.

4.5. Future research directions

This study is the first stage of a series research in examining sustainability of open source software development model. The current research is based on cross-sectional data. In our future research, we plan to design a time series study: we will collection data through open source projects' entire life span. We will pay special attention to the factors that could help a budding project reach a critical mass. This research is based on secondary objective data only. In the future research, we plan to conduct focus group interview and survey on open source developers, especially the core developers.

Another research direction we will pursue in the future is to study the OS software engineering model. We could study such questions as: how specification for projects is gathered, discussed, how the design emerges, and what kind of quality factors is built into the open source coding process.

ACKNOWLEDGEMENTS:

The authors wish to thank the track chair and three anonymous reviewers for their insightful comments and suggestions, which have greatly improved our paper.

REFERENCES:

1. Asundi, J., 2001. Software engineering lessons from open source projects. In Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering. Feller, J., Fitzgerald, B. & van der Hoek, A. (eds).
2. Crowston, K., Annabi, H., & Howison, J. (2003). Defining open source software project success. 24th ICIS Proceedings, pp. 327-340.
3. Fielding, R.Y., 1999. Shared leadership in the Apache project. Communications of the ACM, 42 (4), 42– 43.
4. Free/Libre and Open Source Software: Survey and Study, International Institute of Infonomics, University of Maastricht, The Netherland; Berlecon Research GmbH, Berlin, Germany, June 2002, http://www.infonomics.nl/FLOSS/report/
5. Hann, I., Slaughter, S., Roberts, J., & Fielding, R. (2002). Economic incentives for participating in open source software projects. 23rd ICIS Proceedings, pp. 365-372.
6. Hars and Or, 2002, "Working for Free? Motivation for Participating in Open Source Projects", International Journal of Electronic Commerce, Vol. 6, No. 3.
7. Ioannis S, Ioannis S, Lefteris A, and Apostolos O, 2004, "Open Source Software Development Should Strive for Even Greater Code Maintainability", Communications of the ACM, October, pp. 83-87.

8. Koch, S; Schneider, G, Effort, co-operation and co-ordination in an open source software project: GNOME. Information Systems Journal, Jan2002, Vol. 12 Issue 1, p27, 16p
9. Lerner, J, Tirole, J. 2002. “Some Simple Economic of Open Source”, Journal of Industrial Economics, 50, 197-234.
10. March, J.G. & Simon, H.A., 1958. Organizations. John Wiley, New York.
11. Markus, M.L., Manville, B. & Agres, C.E. 2000. What makes a virtual organization work? Sloan Management Review, Fall, 13–26.
12. Mintzberg, H. 1972. The Structure of Organizations. Prentice Hall, Englewood Cliffs, NJ.
13. Mockus, A., Fielding, R.T. & Herbsleb, J. 2000. A case study of open source software development: the Apache server. Proceedings of the 22nd International Conference on Software Engineering, 263–279
14. Moody, G. 2001. Rebel Code: Linux and the Open Source Revolution. Perseus Press Cambridge, MA.
15. Nohria, N. 1995. Note on organization structure. Harvard Business School, Reprint no. 9–491–083.
16. Oliver, R. 1993. "Cognitive, Affective, and Attribute Bases of the Satisfaction Response", Journal of Consumer Research, Vol. 20, Issue 3, 418-431.
17. Raymond, E. 1999. The Cathedral and the Bazaar.
18. Schmidt, D.C. and Porter, A. 2001. Leveraging open source communities to improve the quality and performance of open source software. In: Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering. Feller, J., Fitzgerald, B. & van der Hoek, A. (eds).
19. Sharma, S. Sugumaran, V. Rajagopalan, B, 2002. A framework for creating hybrid-open source software communities. Information Systems Journal, Vol. 12 Issue 1, p7-26
20. Stewart, K., Ammeter, A., & Maruping, L. (2005). A preliminary analysis of the influences of licensing and organizational sponsorship on success in open source projects. 38th HICSS Proceedings, pp. 1-10.
21. Thomas and Hunt, 2004, Open Source Ecosystems, IEEE Software 32(1)