University of Massachusetts - Amherst
ScholarWorks@UMass Amherst

Schweik Open Source Project | Science, Technology and Society Initiative

12-1-2007

Identifying Success and Abandonment of Free/Libre and Open Source (FLOSS) Commons: A Preliminary Classification of Sourceforge.net projects

Charles M. Schweik
University of Massachusetts - Amherst

Robert English
University of Massachusetts - Amherst

Follow this and additional works at: http://scholarworks.umass.edu/opensource

Part of the Political Science Commons, and the Science and Technology Studies Commons

Schweik, Charles M. and English, Robert, "Identifying Success and Abandonment of Free/Libre and Open Source (FLOSS) Commons: A Preliminary Classification of Sourceforge.net projects" (2007). *Schweik Open Source Project*. Paper 2.
http://scholarworks.umass.edu/opensource/2

This Research, creative, or professional activities is brought to you for free and open access by the Science, Technology and Society Initiative at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Schweik Open Source Project by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

Important Notice: This PDF contains the final version of this article, that was published in the December 2007 issue of Upgrade (vol VIII, no. 6). The previous one, edited by UPGRADE, is an obsolete version.

Identifying Success and Abandonment of FLOSS Commons: A Classification of Sourceforge.net Projects

Robert English
Center for Public Policy and Administration, University of Massachusetts, Amherst, MA USA
renglish@pubpol.umass.edu

Charles M. Schweik
Department of Natural Resources Conservation and Center for Public Policy and Administration, University of Massachusetts, Amherst, MA USA
cschweik@pubpol.umass.edu

Abstract

Free/Libre and Open Source Software (FLOSS) projects are a form of commons where individuals work collectively to produce software that is a public, rather than a private, good. The famous phrase "Tragedy of the Commons" describes a situation where a natural resource commons, such as a pasture, or a water supply, gets depleted because of overuse. The tragedy in FLOSS commons is distinctly different -- it occurs when collective action is abandoned before a software product is produced or reaches its full potential. This paper builds on previous work about defining success in FLOSS projects by taking a collective action perspective. We first report the results of interviews with FLOSS developers regarding our ideas about success and failure in FLOSS projects. Building on those interviews and previous work, we then describe our criteria for defining success or abandonment in FLOSS commons. Finally, we discuss the results and validation of a classification of nearly all projects hosted on Sourceforge.net as of August 2006.

1. Introduction

Free/Libre and Open Source Software projects (FLOSS) are recognized as Internet-based commons [1,13,15]. Since 1968, when the famous article "Tragedy of the Commons" by Garrett Hardin was published in the journal Science, there has been significant interest in understanding how to manage commons appropriately. Hardin's work, and much of the work that followed, focused on commons management in the natural environment. And in these commons, the "tragedy" Hardin described was over-harvesting and destruction of the resource, whether it be water, fish stock, forests, or our atmosphere. In FLOSS commons the "tragedy" is different; what developers hope to avoid is project abandonment and a "dead" project. In order for FLOSS projects to be successful, the collective action involved (or attempts at collective action in the case of projects with one participant) must be sustained at least until a software product has been produced. Discovering how FLOSS projects sustain collective action to produce useful software may have important implications for improving our understanding of FLOSS software development as well as computer-mediated collective action more generally [14,15].

In recent years, scholars have investigated different approaches to measuring the success and failure of FLOSS projects. For example, studies [2,3,7,11,16] measured FLOSS project "life" or "death" by monitoring project activity measures such as: (1) the release trajectory (e.g., movement from alpha to beta to stable release); (2) changes in version number; (3) changes in lines of code; (4) the number of "commits" or check-ins to a central storage repository, and (5) activity or vitality scores measured on collaborative platforms such as SF and Freshmeat.net. Weiss assessed project popularity using web search engines [17]. And most recently, Crowston, Howison and Annabi reviewed traditional models used to measure information systems success and then adapted them to FLOSS [4]. They collected data from Sourceforge.net (SF) and measured community size, bug-fixing time and the popularity of projects.

In this paper, we are trying to build on these studies by defining success and abandonment of FLOSS commons from the perspective of successful collective action. The paper is organized as follows. First, we describe interviews we conducted with FLOSS developers to get feedback on our ideas about defining success. Next, we define a six-stage classification system of FLOSS commons based on information gained from these interviews, as well as previous literature and our own earlier work studying FLOSS. We follow this with a description of our efforts in building a dataset which combines much of the August 2006 data available from

the FLOSSmole project (described below) and data we gathered ourselves through automated data mining of the SF website. This section then describes how we operationalized our proposed success/abandonment classes using this dataset. The "Results" section discusses our preliminary classification of nearly all projects hosted on SF as of August 2006, and the "Validation" section explains how we verified the results. We conclude the paper with some next steps.

2. FLOSS Developer Opinions on Success and Failure

We conducted eight interviews [18] with FLOSS developers between January and May of 2006 in part to get opinions about definitions of success and failure. We stratified our sampling by categories of projects with <5, 5-10, 11-25 and >25 developers and interviewed developers from two projects in each category. Interviews were conducted over the phone, digitally recorded, transcribed and analyzed.

We asked interviewees how they would define success in a FLOSS project. Interviewees responded with five distinct views. One defined success in terms of the vibrancy of the project's developer community. Three defined FLOSS success as widely used software. Two others defined success as creating value for users. One developer cited achieving personal goals, and the last interviewee felt his project was successful because it created technology that percolated through other projects even though his project never produced a useful standalone product.

Immediately after asking interviewees about success, we asked how they would define failure in a FLOSS project. Interestingly, all eight developers said that failure had to do with a lack of users and two indicated that a lack of users leads to project abandonment. In a probing question that followed, we asked if defining a failed project as one that was abandoned before producing a release seemed reasonable. Four interviewees flatly agreed, three agreed with reservations and one disagreed. Two of those with reservations raised concerns about the quality of the release. For example, one project might not make its first release until it had a very stable, well functioning application while another project might release something that was nearly useless. Another interviewee had concerns about how much time could pass before a project was declared abandoned. One developer argued that a project that was abandoned before producing a release could be successful from the developer's point of view if he had improved his programming skills by participating. The dissenting developer felt that project source code would often be incorporated into other FLOSS projects and would not be a failure even if no release had been made.

So, how do these responses inform working definitions of success and abandonment? Because we view FLOSS projects as efforts in collective action with the goal of producing public good software, defining success in terms of producing a useful software product makes sense, and our interviewees seem to agree. Six of the eight interviewees suggested that success involves producing something useful for users. Since the real tragedy for a FLOSS project involves a failure to sustain collective action to produce, maintain or improve the software, defining failure in terms of project abandonment makes sense, and generally, our interviewees agreed. Treating the first release as a milestone or transition point between what we refer to as the "Initiation Stage" and the project "Growth Stage" [13, 18] emerges logically from this line of thinking. All in all, these interviews supported our initial thinking about project success and abandonment.

3. A Success/Abandonment Classification System for FLOSS Commons

After conducting the interviews and considering the results, we developed a six-class system for describing success and abandonment of FLOSS projects across two longitudinal stages of Initiation and Growth (Table 1). In previous work [13, 18] we defined "Initiation" as the start of the project to its first public release, and "Growth" as the period after this release.

Therefore, a project is classified as *(1) Success in the Initiation Stage* (SI) when it has produced "a first public release." This can be easily measured for projects hosted at SF because SF lists all a project's releases. A project that is successful in the initiation phase automatically becomes an indeterminate project in the growth phase.

Projects are classified as *(2) Abandonment in the Initiation Stage* (AI) when the project is abandoned before producing a first public release. We define abandonment as few forum posts, few emails to email lists, no code commits or few other signs of project activity over a one-year period. Preliminary data we have analyzed from SF indicates that projects in Initiation that have not had a release for a year are generally abandoned (see the discussion of the "test sample" below)

A project is considered a (3) *Success in the Growth Stage* (SG) when it exhibits "three releases of a software product that performs a useful computing task for at least a few users (it has to be downloaded and used)." We decided that the time between the first release and the last release must be at least six months because a "growth stage" implies a meaningful time span. As mentioned above, we can easily measure the number of releases and the time between them since SF tracks this information.

Measuring "a useful computing task" is harder and clearly more subjective. Acquiring the number of downloads recorded on project websites is probably the easiest measure, with the assumption that many downloads captures the concept of utility.

A project is considered (4) *Abandoned in the Growth Stage* (AG) when it appears to be abandoned without having produced three releases or when it produced three releases but failed to produce a useful software product.

We classify a project as (5) *Indeterminate in the Initiation Stage* (II) when it has yet to reveal a first public release but shows significant developer activity.

Finally, projects are assigned (6) *Indeterminate in the Growth Stage* (IG) when they have not produced three releases but show development activity or when they have produced three releases over less than six months and show development activity.

4. Operationalizing the Classification System

As a first step in operationalizing our definitions for FLOSS success and abandonment, we conducted a random test sample of sixty projects hosted on SF using April 2005 FLOSSmole data [5]. The FLOSSmole project is itself an open source-like project where researchers and others collaborate to collect and analyze data about FLOSS. The data is collected by automated "crawling" or "spidering" of SF and other open source hosting sites. We decided to conduct this test sample from the FLOSSmole database to look for problems with our classification scheme and to get some idea about the number of projects likely to fall within each of the classes. Following the logic used in our FLOSS developer interviews and knowing we wanted to study projects with larger numbers of developers because of their more interesting collective action issues, we stratified by number of developers into categories of <10, 10-25 and >25 developers. We randomly sampled twenty projects from each category for a total of sixty projects. We chose 20 projects because it was a reasonable undertaking given time constraints. For these sixty sampled projects, we manually compiled data on project registration, last release date, number of downloads, project website URL and forum/email/ postings among other data. From this data, we made a judgment about whether the software was "useful" and whether the project was abandoned. We classified the projects as SI, AI, SG or AG based on this information. No indeterminate cases were found in this sample.

Perhaps the most important information we acquired from the test sample is that the vast majority of projects that have not had a release for a year are abandoned. All 27 projects in the sample that (1) had not provided a release in over a year and (2) had less than three releases were abandoned. This finding suggested that we could produce a relatively simple but approximately accurate classification by using a project's failure to release within a year as a proxy for abandonment.

Naturally, operationalizing the definitions for success and abandonment had much to do with the availability of data. We chose to use the August 2006 data spidered from SF because it was the latest data available at the time. This data has a total of 119,590 projects, but 235 of these projects were missing essential data leaving 119,355 projects. Although FLOSSmole had most of the data we needed for operationalizing our classification, they did not have data on the number of releases and the dates of the first and last release. Consequently, we spidered SF ourselves between September and October 2006 to fill in this data gap. 8,422 projects had missing data or had been deleted from SF (SF occasionally purges defunct projects) between the August 2006 and the time we collected our data leaving valid data for 110,933 projects. Based on our definitions described earlier, and the added information we gained from the test sample, we undertook a classification of these SF projects as described in Table 1.

Table 1: Six FLOSS success/abandonment classes and their methods of operationalization

Class/ Abbreviation	*Definition(D)/Operationalization(O)*
Success, Initiation (SI)	D: Developers have produced a first release. O: At least 1 release (Note: all projects in the growth stage are SI)
Abandonment, Initiation (AI)	D: Developers have not produced a first release and the project is abandoned O: 0 releases AND >=1 year since SF project registration
Success, Growth (SG)	D: Project has achieved three meaningful releases of the software and the software is deemed useful for at least a few users. O: 3 releases AND >= 6 months between releases AND does not meet the download criteria for abandonment detailed in the TG description below.
Abandonment, Growth (AG)	D: Project appears to be abandoned before producing 3 releases of a useful product or has produced three or more releases in less than 6 months and is abandoned. O: 1 or 2 releases and >=1 year since the last release at the time of data collection OR < 11 downloads during a time period greater than 6 months starting from the date of the first release and ending at the data collection date OR 3 or more releases in less than 6 months and >= 1 year since the last release.
Indeterminate	D: Project has no public release but has

Class/ Abbreviation	*Definition(D)/Operationalization(O)*
Initiation (II)	significant developer activity O: 0 releases and < 1 year since project registration
Indeterminate Growth (IG)	D: Project has not yet produced three releases but shows development activity or has produced 3 releases or more in less than 6 months and shows development activity. O: 1 or 2 releases and < 1 year since the last release OR 3 releases and < 6 months between releases and < 1 year since the last release

5. Results

Table 2 provides the number of SF projects classified by the two longitudinal stages: Initation and Growth. It also reports projects that could not be classified. Table 3 summarizes our results of the success and abandonment classification of all projects on SF. As Table 3 column 3 shows, potential classification errors stem primarily from two sources:

Source 1 Error- using one year without a release as a proxy for abandonment.

Source 2 Error - using the number of downloads per month as a proxy for the software being useful.

Table 2: Sourceforge.net projects organized by longitudinal stage (as of August 2006)

Stage	*# of Projects (% of Total classified)*
Initiation Stage	50,662 (47)
Growth Stage	57,085 (53)
Not classified	3,186*
Total classified	107,747
* These are valid projects, but could not be classified because they have 0 releases & downloads on SF but have other websites that may be used for these functions.	

Table 3: Preliminary classification of all FLOSS projects on Sourceforge.net (as of August 2006)

Class	*# of Projects (%of Total)*	*Possible Classification Errors (other than errors in the SF data)*
AI	37,320 (35)	The project is not abandoned but > 1 year old
SG	15,782 (15)	The software is not used in spite of not meeting the download criteria for abandonment
AG	30,592 (28)	The project is not abandoned; OR
		The project produced useful software even though it met the download critera for abandonment
II	13,342 (12)	No classification errors (by definition)
IG	10,711 (10)	No classification errors (by definition)
Total	107,747	
Note: SI is not listed because these successes are Growth Stage projects. Including SI would double count.		

6. Validation of Results

To test the validity of the results in Table 3, we took a random sample of three hundred classified projects, and checked each projects' classification results by manually reviewing SF pages. Table 4 lists validation results.

Table 4: FOSS Project Classification Validation Results

Original Class (# of cases)	Correct	In-correct	Deleted/ Missing Data	Error Rate %
AI (106)	77	10	19	11.5
AG (101)	93	8	0	7.9
SG (93)	92	0	1	0
Totals	262	18	20	6.4

Of the 106 projects originally classified AI, 77 were correctly classified, ten were incorrectly classified, eighteen were deleted from SF and one had missing information and could not be validated, resulting in the highest classification error rate of 11.5%. The ten misclassifications did not list a release for a year after they were registered, but did show some developer activity in the year before our data were collected (Source 1 error). Regarding the eighteen deleted projects, it is highly likely that most if not all of these were classified correctly, given SF regularly purges inactive projects; however, it is possible that some were active and were moved to other hosting platforms by the project developers. Consequently, we keep 11.5% as the error rate for AI.

Of the 101 cases that were originally assigned to the AG class, eight were active and incorrectly classified for an error rate of 7.9%. Finally, of the 93 cases that were classified as SG, 92 were classified correctly and one could not be validated because of missing data on SF.

These validation results show that the classification varies from what we would consider "reasonably accurate" (AI) to "extremely accurate" (SG). This gives a high-level of confidence that future analysis based on this classification will produce meaningful results.

7. Conclusion

We intend to use this classification as a dependent variable for quantitative models that investigate factors that lead to success and abandonment in FLOSS in the stages of Initiation and Growth. We expect influential factors to be different between these two stages [13, 18].

The validation results give us confidence that this classification is reasonably accurate, despite the shortcomings described in Section 5. We are publishing these definitions and results in the spirit of "release early and often" and because defining and classifying success in FLOSS projects is important to many FLOSS research projects. In the near future, we plan to release the data we collected and our classifications on the FLOSSmole site. We hope that in the tradition of open source collaboration other researchers will build on this work by correcting any perceived "bugs" in our approach and collecting additional data to improve classification accuracy.

8. Acknowledgments

Support for this study was provided by a grant from the U.S. National Science Foundation (NSFIIS 0447623). However, the findings, recommendations, and opinions expressed are those of the authors and do not necessarily reflect the views of the funding agency. Thanks go to Megan Conklin, Kevin Crowston and the FLOSSmole project (http://ossmole .sourceforge.net/) for making their Sourceforge data available, and for their assistance. We are also grateful to Thomas Folz-Donahue for programming work building our FLOSS project database.

9. References

[1] Bollier, D., *Silent Theft: The Private Plunder of Our Common Wealth*, Routledge, London, 2002.

[2] A Capiluppi, P. Lago, and M Morisio,. "Evidences in the Evolution of OS projects through Changelog Analyses," In J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (eds.) *Taking Stock of the Bazaar: Proceedings of the 3rd Workshop on Open Source Software Engineering*, 12 Dec. 2006; http://opensource.ucc.ie/ icse2003.

[3] K. Crowston, H. Annabi, and J. Howison, "Defining Open Source Project Success," In *Proceedings of the 24th Int.l Conference on Information Systems*, ICIS, Seattle, 2003.

[4] K. Crowston, J. Howison, H. Annabi, "Information Systems Success In Free And Open Source Software Development: Theory And Measures," *Software Process Improvement and Practice*, v 11, n 2, March/April, 2006, pp. 123-148.

[5] FLOSSmole, "sfProjectInfo06-Apr-2005," 16 June 2005; http://sourceforge.net/project/showfiles.php?group_id=119453 &package_id=132043/.

[7] S. Hissam, C. B. Weinstock, D. Plaksoh, and J. Asundi,. *Perspectives on Open Source Software. Technical report CMU/SEI-2001-TR-019*, Carnegie Mellon University. 10 Jan. 2007,http://www.sei.cmu.edu/publications/documents/01.reports /01tr019.html.

[11] M. Robles, G. Gonzalez,- J.M. Barahona, J. Centeno-Gonzalez, V. Matellan-Olivera, and L. Rodero-Merino, "Studying the Evolution of Libre Software Projects Using Publically Available Data," In J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (eds.) *Taking Stock of the Bazaar: Proceedings of the 3rd Workshop on Open Source Software Engineering*, 12 Dec. 2006. http://opensource.ucc.ie/icse2003.

[13] C. Schweik, "An Institutional Analysis Approach to Studying Libre Software 'Commons'", *Upgrade: The European Journal for the Informatics Professional*, 10 Jan. 2007, http://www.upgrade-cepis.org/issues/2005/3/up6-3Schweik.pdf.

[14] C. Schweik, T. Evans and J. Grove, "Open Source and Open Content: A Framework for Global Collaboration," in *Social-Ecological Research. Ecology and Society* 10 (1): 33. 10 Jan. 2007, http://www.ecologyandsociety.org/vol10/iss1/ art33/.

[15]C. Schweik, "Free / Open Source Software as a Framework for Scientific Collaboration," In Hess, Charlotte, and Elinor Ostrom, eds. *Understanding Knowledge as a Commons: From Theory to Practice*, MIT Press, Cambridge, Mass, 2007.

[16] K. J. Stewart, and T. Ammeter, "An Exploratory Study of Factors Influencing the Level of Vitality and Popularity of Open Source Projects," In L. Applegate, R. Galliers, and J.I. DeGross (eds.) *Proceedings of the 23rd International Conference on Information Systems*, Barcelona, 2002, pp. 853-57.

[17] D. Weiss, "Measuring Success of Open Source Projects Using Web Search Engines," *Proceedings of the First International Conference on Open Source Systems, Genova, 11th-15th July 2005*. Marco Scotto and Giancarlo Succi (Eds.), Genoa, 2005, pp. 93-99

[18] C. Schweik and R. English, ""Tragedy of the FOSS Commons? Investigating the Institutional Designs of Free/Libre and Open Source Software Projects," *FirstMonday*. 28 Feb. 2007, http://www.firstmonday.org/issues/issue12_2/schweik/.