The Dependent Variable: Defining Open Source "Success" and "Abandonment" Using Sourceforge.Net Data

The Dependent Variable:
Defining Open Source "Success" and "Abandonment" Using Sourceforge.Net Data

Charles M. Schweik

National Center for Digital Government
Center for Public Policy and Administration
Department of Natural Resources Conservation
University of Massachusetts, Amherst

NCDG Working Paper No. 09-003

Submitted December 7, 2009

The National Center for Digital Government is supported by the National Science Foundation under Grant No. 0131923. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Support for "The Dependent Variable" was provided by a grant from the U.S. National Science Foundation (NSFIIS 0447623). Any opinions, findings, conclusions, or recommendations expressed in this material are the authors and do not necessarily reflect the views of this agency.

PART III

EXPLORATORY DATA ANALYSIS OF SOURCEFORGE.NET DATA

INTRODUCTION TO PART III

In Part II of this book, we explored theoretical concepts involved in explaining the open source software phenomenon. In this section, Part III, we undertake an empirical examination of open source projects to investigate the research questions and hypotheses presented in tables at the conclusions of Chapters 2, 3, 4 and 5, respectively.

To do this, we take advantage of a very large database of projects (107,000+) found on the open source hosting site Sourceforge.net. This database was compiled by people involved in the FLOSSMole project (2006) along with some data we collected from Sourceforge.net ourselves. As we emphasize in Chapter 6, we see Sourceforge.net as a kind of "remote sensor" of open source projects, analogous to satellites like Landsat that monitor our Earth. Given that Sourceforge.net will likely be around for some time, it is worthwhile to try and understand what this database can tell us about open source collaboration.

Our first task is to utilize Sourceforge.net project data to develop a theoretically sound and robust measure of open source collaborative success or abandonment – our dependent variable. We present this work in Chapter 6. In Chapter 7, we describe the independent variables we use in our analyses, and review the Classification Tree statistical methods we use. With this foundation in place, we examine the factors that help to distinguish between successful or abandoned Sourceforge.net projects in the two longitudinal stages we introduced in Chapter 3 (Figure 3.2): the "Initiation" or pre-release stage (Chapter 8), and the "Growth" or post-first release stage (Chapter 9). Each chapter reports its own findings for the stage it analyzes. We close Part III of the book with a comparative discussion of differences and similarities between these two stages. We also emphasize (as anticipated) that the SF database is incomplete to capture all the theoretical factors thought to explain open source success and abandonment.

CHAPTER 6

THE DEPENDENT VARIABLE: DEFINING OPEN SOURCE "SUCCESS" AND "ABANDONMENT" USING SOURCEFORGE.NET DATA[1]

From the very beginning of this research project, we understood that we needed to define what success meant in open source so that we could use that definition to create a dependent variable for our empirical studies. Does success mean a project has developed high quality software, or does it mean that the software is widely used? How might extremely valuable software that is used by only a few people, such as software for charting parts of the human genome, fit into this definition? In this chapter, we establish a robust success and abandonment measure that satisfies these conditions. We describe the process we went through to create a definition of open source success and abandonment, and how we used that definition to classify nearly all the projects hosted on Sourceforge.net (SF, as of October 2006) as either successful or abandoned.

The work to create a defensible measure of project success and abandonment was time-consuming and extensive. It involved interviewing open source developers (a different set of interviews than the ones we will present later in Chapter 10), studying the work of others (e.g., Capiluppi, Lago, & Moriso, 2007; Crowston, Annabi & Howison, 2003; Hissam, Weinstock, Plaksoh & Asundi, 2007; Robles et al., 2007; Stewart and Ammeter, 2005), and doing test sampling of SF data. Our resulting classification is, as far as we can tell, the first comprehensive success and abandonment classification of SF projects, and with it established, we can answer simple, but previously unanswered questions like: How many

1 This chapter is based on material we previously published in *Upgrade: The European Journal for the Informatics Professional (*http://www.upgrade-cepis.org). We are grateful to the *Upgrade* editors for granting us permission to do so.

projects hosted on SF continue to be worked on collaboratively? How many are abandoned? Are projects using particular programming languages or in certain topic areas of software development more likely to achieve collaborative success? Or, are projects targeted at particular audiences or being developed for open source operating systems more likely to succeed? The data stored on SF are important because they hold the answers to these and other similar questions.

The next section of this chapter describes SF in more detail, along with the types of data recorded by SF, and our ideas about how representative SF is of open source in general. We then describe our interviews with open source software developers that were designed to collect feedback about our initial ideas about how to define open source project success. Later sections of this chapter describe our six-stage success and abandonment classification system and the methodology we used to classify SF projects. We developed a definition of project success that is based on successful collective action rather than high quality or widespread use. The chapter closes with a discussion of our results and the statistical validation of those results, as well as our conclusions about the advantages and deficiencies of success and abandonment as our dependent variable.

About Sourceforge (SF)

We have mentioned the SF open source project web hosting site in earlier chapters, but before now, we have never described it in detail. As of this writing (July 2009), SF has over 2 million registered users and provides free-of-charge web hosting services and collaborative tools for over 230,000 open source software projects (Sourceforge, 2009). The tools provided by SF include: software code repositories (with version control), bug-tracking

utilities, online forums, email mailing lists, a wiki, links to a separate project website in addition to the project's presence on the SF website, file downloading services for the project's software, and a web-based administrative interface so that open source software project administrators can manage all of these tools. In short, SF provides the tools that most projects require to develop open source software. The oldest projects currently hosted on SF were registered in 1999 when the service started.

Data Available for Projects Hosted on SF

SF stores and makes publicly available much of the information generated by a project's administrator when the project is created, as well as any information generated over time by the tools the project uses. For example, the project's pages on SF provide the date the project was registered and the number of developers working on the project, as well as the number of times the project's code repository has been accessed, historical records of the number of bug posts and feature requests, searchable archives of forum posts and emails, the number of project software releases, and the total number of times the software has been downloaded. And, this only describes a portion of the information available for a given project. We will describe these data in more detail later in this chapter and in Chapter 7.

The people associated with two academic projects – one at the University of Notre Dame called the Sourceforge Research Data Archive (Van Antwerp and Madey, 2008) and the other, the FLOSSmole project (FLOSSmole, 2005) based at Syracuse University (http://ossmole.sourceforge.net/) – understand the importance of capturing and keeping historical SF data. For researchers like us who are trying to learn about and explain the open source development phenomenon, the availability of these historical repositories is critically important. We view these projects as “remote sensors” of open source project activity.

Through their efforts to periodically sample or collect temporal "slices" of the SF database, these projects are actively building a historical repository of open source projects similar to the way NASA's Landsat satellite system and the EROS Space Center have collected and archived historical data about change in conditions of the earth's surface. There are great analytical opportunities in these datasets. Although we only look at one timepoint of the SF database in the empirical analysis of Sourceforge projects we will present in Chapter 7, we believe our analysis captures a time-series element because the dataset is a snapshot of approximately 107,000 projects all at different points in their project life cycle. Given the likelihood that SF will be around for years to come (and hopefully these archival projects will be too), it is important to investigate what can be learned about open source using SF data alone. This chapter, along with the rest of Part III of this book, is devoted to this task.

Is Sourceforge Representative of all Open Source Projects?

Because our classification of SF projects and our analysis in Chapters 8 and 9 exclusively use data gathered from SF, it is important to ask the question: to what degree is SF representative of the population of open source projects "out there"? To what degree would any findings based on SF apply to open source in general? In our view, the representativeness of SF can be defined along at least two dimensions: First, whether or not the projects on SF are representative of the broader universe of open source projects, and second, whether or not developers on SF representative of the broader universe of open source developers.

While SF hosts the greatest number of open source projects, as we noted in Chapter 1, there are many other open source project hosting sites, including BerliOS, Freepository, Savannah, Sharesource, and others (see Open Directory Project, 2008). In addition, many

open source projects maintain their own web sites and other project infrastructure on their own servers. Based on an Internet search and literature review (see Appendix 6.1 for search procedures), it appears that no one has a sound estimate of how many Free/Libre and Open Source Software projects exist or how many people are working on them. Since the population of open source projects and developers is unknown, it is not surprising that we were not able to find any empirical analysis to assess how representative SF is of this unknown population.

However. researchers have considered this or closely-related questions. In one of the earliest references to the representativeness of SF, Madley, Freeh and Tynan (2002) mention that they assumed that SF was representative because of its popularity and because of the number of projects hosted there, although they note that this assumption "needs to be confirmed" (p. 1812). Confirming their assumption will prove difficult because it requires determining the extent of the open source universe – which, as we mentioned above, has not yet been accomplished. For example, Ghosh (2005) notes that there is no census of the universal population of open source developers. Additionally, there does not appear to be any estimate of the total number of open source projects, although Carlo Daffara (2007), who is associated with the European Union's European Working Group on Libre software, helped to create an estimate of the number of "active and stable" projects for the FLOSSMetrics (2007) project. Daffara defined active projects as projects with an 80-100% activity index (SF and some other hosting sites generate an "activity" index) and a release within the last six months. In the process of arriving at their estimate, Daffara (2007) examined 100 projects each from Sourceforge, BerliOS, Savannah, Gna and other hosting sites which showed that active projects comprise about 8-10% of the total. By using various techniques and information from other sources, they arrived at a *lower-bound* figure of 195,000 projects in the open source

universe, with about 13,000 active projects and about 5,000 active, mature and stable projects. Because they were only interested in active and stable projects that were useful for Small and Medium- Sized Enterprises (SMEs), they did not try to estimate an *upper-bound* figure for the population of open source projects.

With regard to the second representative dimension, "developers," Ghosh (2005) states that the FLOSS survey was random and thus the results of the survey are applicable to the entire universe of open source developers. One way to help establish whether SF is representative would be to duplicate Ghosh's FLOSS survey on a random sample drawn from SF developers. If the results are statistically similar, then this would support the hypothesis that SF developers are representative of the universe of open source developers. While to our knowledge, no one has conducted such a study. However, Raymond (2004) did a Web search on the use of the terminology "open source" versus the terminology "free software[2]" on both SF and the Internet as a whole. Raymond found that about 3% of the developers on SF use the term "free software" and that about 4.5 % of the developers on the Web in general use the term "free software," while about 99% of the developers on Savannah.gnu.org (a hosting site set up by "free software" advocates) used the term "free software." Although this does not make a strong case – it represents only one (philosophical) attribute of software developers (refer back to Chapter 3) – it gives a sense that the developers in SF use philosophical terminology roughly similar to the more broad population of open source developers found across Internet.

Spaeth et al. (2007) argue that sampling the Debian GNU/Linux Distribution is more

2 Recall from Chapter 1 that these terms reflect a philosophical difference, where "free" or "libre" software captures the viewpoint that software should be a public good, whereas "open source" software is sometimes considered more open or willing to work with commercial interests. Recall also that in this book for ease of reading we use the "open source" widely to include both philosophies.

representative for some purposes than sampling SF for two reasons: (1) because "a distribution...represents the population of projects in use," and (2) because Debian includes projects hosted on several open source hosting sites in addition to projects that host their own projects on their own hardware. A "distribution" is a release of an operating system, often Linux, combined with a large number of commonly used software programs, and thus, Spaeth refers to a distribution as being representative of open source projects that are actually used by a large number of people. Because Debian selects projects from a number of hosting sites, and from projects that are hosted privately, Spaeth feels that this diversity may be more representative than a single hosting site like SF.

However, Spaeth et al. (2007) also point out that projects associated with Debian must have a "free" license. In addition, all associated Debian free software projects will be successful by our definition (see our definitions below) in order to be included in the Debian package. Furthermore, software that runs on Windows is not included in the Debian Distribution. From this, we conclude that Debian is not representative of the entire population of open source projects because projects with non-free licenses, failed projects, and projects that run on operating systems other than Linux are not included.

Many reasons exist for believing that the projects hosted on SF are more representative of the open source universe than any other repository. First, SF hosts more projects and users than any other repository. Second, Daffara (2007) shows that the rate of active projects on SF is very nearly the same as other hosting sites, thus providing a piece of evidence that SF is representative of other hosting sites. Third, Raymond's (2004) web search provides evidence that the developers on SF are roughly representative of the developers on the entire Internet, at least in terms of their use of certain terminology. Finally, using the data we have (FLOSSMole, 2006) SF has over one thousand projects having one hundred

thousand downloads or more and over 147 projects having over one million downloads. If Daffara (2007) is correct that only about 5,000 stable mature and active projects exist in the entire population of open source projects, then it is likely that many of them are represented on SF.

One of the reviewers of an earlier version of this book said: “there is often an unstated assumption [in open source research] that there is such an animal as ‘The Open Source Developer’ and that he/she lives in SourceForge – and neither is true. There are dozens of radically different development contexts many highlighted in the previous chapter and each context is populated with communities of developers with very different motivations and characteristics.” We agree. Chapter 3 emphasized the heterogeneity in developer types, and we recognize that there are a variety of contexts (project hosting sites or self-hosted collaboration platforms) where open source developers work. But there is evidence that suggests that SF is representative of other hosting sites, and has more projects, more failed projects and a significant number of active, mature and stable projects (Daffara, 2007; English and Schweik, 2007). SF may be the best single source of data representative of the entire open source population and consequently, provides a good place for investigating the questions we posed in Chapters 2-5.

With the significance and utility of SF noted, we can move to the discussion of how we used some of these data to create a measure of the success of open source projects. But first, let us take a look at what some open source developers we interviewed thought about success and abandonment of projects.

Developer Opinions about Success and Abandonment

We conducted eight interviews (Schweik and English, 2007) with open source developers between January and May of 2006 to get opinions about definitions of success and, at that time, we were using the term “failure” rather than abandonment. 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 an open source project. Interviewees responded with five distinct views. One defined success in terms of the vibrancy of the project’s developer community. Three defined open source 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 an open source 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 agreed outright, 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 failed or 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 open source projects and would not be a failure even if no release had been made. These discussions led us toward the use of the term of "abandonment" rather than "failure" because many projects that had ceased collaborating still would not be viewed as failed projects.

So, how do these responses inform working definitions of project success and abandonment? Because we view open source projects as a commons driven by collective action with the goal of producing software (recall Chapter 2), defining success in terms of producing "useful software" makes sense. Six of the eight interviewees suggested that success involves producing something useful for users. Since the real "tragedy of the commons" for an open source project (see Schweik and English, 2007) 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" (see Figure 3.2; Schweik, 2007; Schweik and English, 2007) emerges logically from this line of thinking.

A Success/Abandonment Classification System for Open Source Commons

In recent years, scholars have investigated different approaches to measuring the success and failure of open source projects. For example, studies such as Capiluppi, Lago, and Moriso (2007), Crowston, Annabi and Howison (2003), Hissam, et al. (2007), Robles et al. (2007), and Stewart and Ammeter (2005) measured open source 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 software code repository, and (5) activity or vitality scores measured on collaborative platforms such as SF and Freshmeat.net. Weiss (2005) assessed project popularity using web search engines. Crowston, Howison and Annabi (2006) reviewed traditional models used to measure information systems success and then adapted them to open source. They collected data from Sourceforge.net (SF) and measured community size, bug-fixing time and the popularity of projects.

After conducting our interviews, reviewing the ideas and work of the other researchers above, and carefully considering those inputs along with our own thinking about success and abandonment from a "commons" and collective action perspective, we developed a six-class system for describing success and abandonment of open source projects across our two longitudinal stages of Initiation and Growth (see Table 6.1). Recall that in Chapter 3 (see Figure 3.2), we defined "Initiation" as the start of the project to its first public release, and "Growth" as the period after this release.

We classify a project as (1) *Successful in the Initiation Stage (SI)* when it has produced "a first public release." This can be easily measured for SF lists all of 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) *Abandoned 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 classified as 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 it needs to capture some relatively significant effort. (We have found in some cases on SF multiple releases are posted over a single day which would in relative terms not be a meaningful new release.) As mentioned above, we can easily measure the number of releases and the time between them, since SF tracks this information. However, measuring “a useful computing task” is harder and 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.

Table 6.1 Our Dependent Variable: Six Success/Abandonment Classes and their Methods of Operationalization	
Class/Abbreviation	**Definition(D)/Operationalization(O)/SF Variables used(SFV)**
Success, Initiation (SI)	**D:** Developers have produced a first release. **O:** At least 1 release (Note: all projects in the growth stage are SI) **SFV:** Number of Releases
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 **SFV:** Number of Releases, Project Lifespan
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 has >10 downloads. **SFV:** Number of Releases, First Release Date, Last Release Date, Downloads, Data Collection Date
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 3 or more releases and < 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. **SFV:** Number of Releases, First Release Date, Last Release Date, Data Collection Date, Downloads, Project Lifespan
Indeterminate Initiation (II)	**D:** Project has yet to reveal a first public release but shows significant developer activity. **O:** 0 releases and < 1 year since project registration **SFV:** Number of Releases, Project Lifespan
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 **SFV:** Number of Releases, First Release Date, Last Release Date, Data Collection Date

Operationalizing the Classification System

As a first step in operationalizing our definitions for open source success and abandonment using the definitions (denoted as "D:") in Table 6.1, we conducted a random test sample of sixty projects hosted on SF using the FLOSSmole project data we referred to earlier (FLOSSmole, 2006). The FLOSSmole project is itself an open source-like project

where researchers and others collaborate to collect and analyze data about open source software. The data is collected by automated "spidering[3]" of SF and other open source hosting sites, and made publicly available. 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 open source 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, similar to the coding often done in qualitative case study methods, 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 this test sample effort 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. This test sample process and qualitative analysis provided

3 For non-technical readers, this means writing a program that reads pages on these websites and extracting out the data needed.

confidence that our conceptual ideas for these definitions were accurate. Our next step was to implement these concepts using SF data in our dataset.

Data Description

Of course, the operationalization of the definitions for success and abandonment have much to do with the availability of data in SF to implement these concepts. We needed a number of variables to operationalize our classification system. These variables included: Project Lifespan, Number of Releases, First Release Date, Last Release Date, Data Collection Date and Downloads. Fortunately, many of these variables existed in data gathered by FLOSSmole. We used FLOSSmole data from August, 2006 because this was the most recent data available at the time we developed this classification. However, not all of the data we needed were available from FLOSSmole, so we spidered the SF website ourselves between September 24, 2006 and October 16, 2006 to gather the missing variables. The missing variables included: Number of Releases, First Release Date and Last Release date. We call the data we spidered ourselves the "UMass data."

The FLOSSmole data had complete data for 119,355 projects, but 8,422 of these projects either had missing data or were purged from SF between the time the FLOSSmole data were collected and the time the UMass data were collected, thus leaving 110,933 projects. We eliminated another 3,186 projects from our classification because these projects had zero releases and downloads listed on SF, but also had project websites not hosted on SF that may have been used to distribute files.[4] In the end, we classified 107,747 projects.

[4] In later analyses, had we left these projects with external project web pages in, they could have been falsely classified as abandoned even though they were active in other web locations than Sourceforge.

Our dataset also includes other numerical and categorical independent variables for these projects. These are used to construct both independent and dependent variables. Independent variables (and the associated approach to analysis) will be described in Chapter 7. The variables used to create our dependent variable are described more precisely below, and Table 6.2 shows descriptive statistics for these variables.

Project Lifespan

Project Lifespan is the time between the date the project was registered on SF and the time our data were collected. Since the UMass data were collected after the FLOSSmole data, the data collection dates for the UMass data were used in our calculations.

Number of Releases

The Number of Releases variable is somewhat complicated by the fact that SF lists information for “Package,” “Release,” and “Filename” on project download pages. Projects can release one or more “Packages” that can each contain one or more “Releases.” In addition, the “Releases” can each contain one or more “files.” SF totals both the number of “releases” and the number of “files released” at the bottom of each project's download page. These totals sum the number of releases and files in all packages. The number of “files released” is often greater than the number of “releases.” In our dataset, we created the “Number of Releases” variable which is the total number given for SF “Releases,” rather than using the total number of “files released.” If there are several files released within a “release,” quite often they are help files or different file formats for the same working program, so using “files released” would not be appropriate for the purposes of our classification.

First Release Date

This variable is the date the first release was made.

Last Release Date

This variable is the date the last release was made.

Data Collection Date

This variable is recorded by the spidering software when the web page is downloaded from SF.

Downloads

Our downloads variable is the total number of downloads over the project lifetime for each project.

Table 6.2
Descriptive Statistics for Dependent Variable Components: FLOSSMole (2006) and UMass Sept-Oct 2006 Spidered Data.

Variable Name	Min	1st Quad	Median	Mean	3rd Quad	Max
Project Lifespan (yrs)	0.003	1.08	2.39	2.54	3.70	6.74
Number of Releases	0	0	1.00	2.77	2.00	537
Downloads	0	0	23	12,835	494	228,643,712

Classification Results

Table 6.3 provides the number of SF projects classified in each of our two longitudinal stages: Initiation and Growth using the FLOSSMole (2006) and UMass spidered data. In these 107,747 SF projects, about half were in the Initiation Stage and the other half were in the Growth Stage. Table 6.3 also reports projects that could not be classified.

Table 6.3
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 6.4 summarizes our results of the success and abandonment classification of all projects on SF. As Table 6.4 column 3 shows, potential classification errors stem primarily from two sources: (1) Source 1 Error – using one year without a release as a proxy for abandonment, and (2) Source 2 Error – using the number of downloads per month as a proxy for the software being useful.

Table 6.4 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)**
Abandoned in Initiation (AI)	37,320 (35)	The project is not abandoned but > 1 year old
Successful Growth (SG)	15,782 (15)	The software is not used in spite of not meeting the download criteria for abandonment
Abandoned in Growth (AG)	30,592 (28)	The project is not abandoned, OR the project produced useful software even though it met the download criteria for abandonment
Indeterminate in Initiation (II)	13,342 (12)	No classification errors (by definition)
Indeterminate in Growth (IG)	10,711 (10)	No classification errors (by definition)
Total	107,747	
Note: SI is not listed because these successes are now Growth Stage projects. Including SI would double count.		

Validation of Results

To test the validity of the results in Table 6.4, we took a random sample of three hundred classified projects, and checked each project's classification results by manually reviewing its SF pages. Table 6.5 lists validation results. Of the 106 projects originally classified Abandoned in Initiation (AI), 77 were correctly classified, ten were incorrectly classified, eighteen were deleted from SF and one had missing information and could not be validated (19 missing or deleted in total), resulting in our 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, but the true error rate is probably lower. Of the 101 cases that were originally assigned to the Abandoned in Growth (AG) class, eight were active and incorrectly classified for an error rate of 7.9%. Finally, of the 93 cases that were classified as Successful in Growth (SG), 92 were classified correctly and one could not be validated because of missing data on SF. In other words, our SG classification had an error rate of very close to 0. 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 we can use this measure as a dependent variable in the analysis we present in Chapter 7 and later in the book.

Table 6.5
FOSS Project Classification Validation Results

Original Class (# of cases)	Correct	Incorrect	Deleted or 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 (300)	262	18	20	6.4

Conclusion/Summary

This chapter is critical for this study for it explains how we conceptualized and produced the dependent variable – success and abandonment of SF projects – that is used in much of the empirical work that follows. First, we explained how data is produced and stored on the SF web site, and how representative that data may be of open source projects in general. Although we cannot know for sure whether SF is truly representative of the open source phenomenon as a whole, we believe it is the most representative single source of data available at this time. Next, we described our efforts to come up with a defensible definition for open source project success and abandonment. We did this by conducting interviews with developers that revealed their ideas about these concepts and reviewing existing work done on this subject. We presented our definitions (Table 6.1) and tested these definitions by conducting a test sample of SF projects, classifying them based on our ideas, and manually reviewing the projects. These activities helped finalize our ideas about how to define and operationalize success using SF data. We then discussed how we operationalized these concepts using SF data. In the end, we have created a classification system based on successful collective action as opposed to a classification system that classified successful projects from a software engineering point of view (e.g., a measure of software quality) or the concept of a "large number of users" (recall our classification treats a small number of users (e.g., specialized software) as a successful case. Finally, we presented the results of our efforts to validate the classification, and based on those results, we believe that this classification, used as a dependent variable, has adequate accuracy to produce meaningful results in statistical analyses of SF data.

All the data used in our classification and more detailed working notes on our

classification process have been given back to the FLOSSMole project, and are available on our project website (http://www.umass.edu/digitalcenter/ossuccess/). We hope that other researchers will consider the use of this classification definition – and, in the spirit of open source build and improve upon it – to classify projects at various points in time other than August, 2006. With our dependent variable in hand, we can now analyze how project characteristics affect success and abandonment, and begin to investigate the question: What can SF data tell us about what makes open source projects a successful collaboration? In Chapters 7, 8 and 9 we investigate this question.

Appendix 6.1

Search Procedure for Literature on the Representativeness of Sourceforge.net (Summer, 2007)

GOOGLE

Phrase:

population open source software project - 255,000 results narrowed search
"population of open source projects" - 2 results looked at both
population "open source projects" - 39,600 results looked at top ten
estimate population "open source" project - 56,500 results looked at 20
"estimate population" "open source projects" - 11 results looked at all
is sourceforge.net representative of open source population? - 33,100 results looked at top ten
total number open source software projects - 3,190,000 results (found Galoppini's estimate)
"total number of open source software projects" - 0 results
estimate "total number" "open source projects" - returned 3,370 looked at top 20
census of open source software projects - 1,170,000 results
census "open source projects" - 5,990 results looked through 100
is sourceforge.net representative of open source population? - 5,230 results

GOOGLE SCHOLAR

Phrase:

population open source software project - 124,000 results
"population of open source projects" - 0 results
population "open source projects" - 997 results
estimate population "open source" project - 10,200 results looked at 20
"estimate population" "open source projects" - 0 results
total number open source software projects - 237,000 results looked at top 20
"total number of open source software projects" - 0 results
estimate total number open source projects - 233,000 results looked at 100
estimate "total number" "open source projects" - 341 results looked at 150
census of open source software projects - 29,300 results
census "open source projects" - 97 results looked through 30

Found http://www.osscensus.org/

http://waughpartners.com.au/research/census2007

is sourceforge.net representative of open source population? - 934 results.(Madley paper 2002)

SEARCHED "ACADEMIC SEARCH PREMIER"

Phrase:

population open source projects - 0 results
census open source projects - 0 results
survey open source projects - 0 result
estimate number open source software project - 0 results
sourceforge - 26 results looked at 26

SEARCHED ENGINEERING VILLAGE
Phrase:
population open source projects - 27 results looked at 27
census open source software project - 1 result
survey open source projects - 64 results
estimate number open source software project - 6 results
sourceforge - 72 results
sourceforge representative - 4 results looked at 4

SEARCHED IEEE XPLORE
Phrase:
population open source projects - 0 results
census open source software project - 0 results
survey open source projects - 0 results
estimate number open source software project - 0 results
sourceforge - 16 results looked at 16

SEARCHED ISI WEB OF KNOWLEDGE
Phrase:
population open source projects - 3 results
census open source software project - 0 results
survey open source projects - 20 results looked at 20
estimate number open source software project - 0 results
sourceforge - 72 results
sourceforge representative

SEARCHED SCIENCE DIRECT
Phrase:
population open source projects - 4 results
census open source software project - 0 results
survey open source projects - 17 results looked at 17
estimate number open source software project - 1 result
sourceforge - 13 results