2009

Factors Leading to Success or Abandonment of Open Source Commons: An Empirical Analysis of Sourceforge.net Projects

Charles M Schweik, *University of Massachusetts - Amherst*
Robert English
Sandra Haire

Factors Leading to Success or Abandonment of Open Source Commons: An Empirical Analysis of Sourceforge.net Projects

Charles M. Schweik[1], Robert English[2] , Sandra Haire[3]

[1]Dept Natural Resources Conservation, Center for Public Policy and Administration, National Center for Digital Government, University of Massachusetts, Amherst, USA, cschweik@pubpol.umass.edu

[2] National Center for Digital Government, University of Massachusetts, Amherst, USA, renglish@gmail.com

[3]Dept Natural Resources Conservation, University of Massachusetts, Amherst, USA, shaire@nrc.umass.edu

Preprint version. Published version available at

Schweik, C., English, R. and Haire, S. 2009. Factors Leading to Success or Abandonment of Open Source Commons: An Empirical Analysis of Sourceforge.net Projects. *South African Computer Journal*. Vol. 43. http://journals.sabinet.co.za/comp/

Abstract

Open source software is produced cooperatively by groups of people who work together via the Internet. The software produced usually becomes the "common property" of the group and is freely distributed to anyone in the world who wants to use it. Although it may seem unlikely, open source collaborations, or "commons," have grown phenomenally to become economically and socially important. But what makes open source commons succeed at producing something useful, or alternatively, what makes them become abandoned before achieving success? This paper reviews the theoretical foundations for understanding open source commons and briefly describes our statistical analysis of over one-hundred-thousand open source projects. We have found, that leadership, clear vision and software utility may be causes of success early in a project's lifetime, and that building a community of software developers and users around an increasingly useful software product appears to be key to success for most projects later in their lifetimes.

1. Introduction

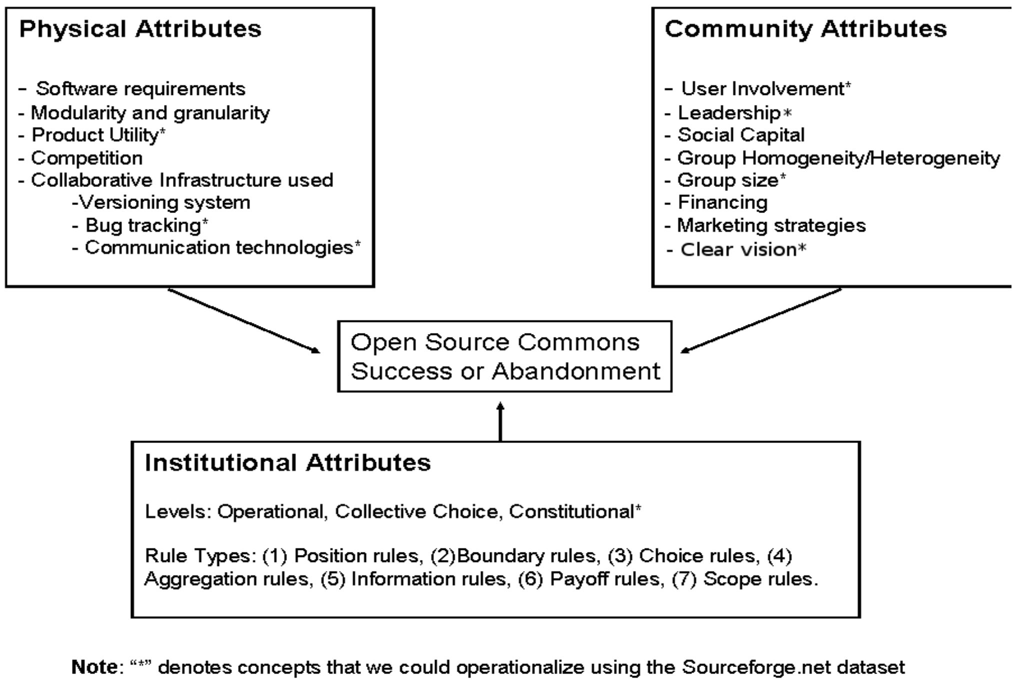
We began to study open source software collaborations, or what we call open source “commons,” in 2005 with support from a United States National Science Foundation grant.[1] Our motivation was to understand these kinds of collaborations to help inform future software projects, but also because the collaborative principles of open source can be applied to other areas where there is a need, globally, to solve pressing problems facing humanity, such as in the case of addressing climate change. Since we began our research, open source commons have become more economically important, possibly affecting 4% of European GDP by 2010 [9]. In addition, peer production commons organised around digital content other than software, like Wikipedia and YouTube, have had significant social effects. Consequently, understanding how these commons work is becoming increasingly important for understanding how our economies and societies may change in the future.

In order to understand open source commons, we needed to know what factors might make these collaborations succeed or be abandoned before reaching their potential. To learn about these factors, we initially looked at the relevant academic literature and developed hypotheses about what might be the important factors. We then undertook empirical research, including interviews with open source software developers and statistical data analysis of projects hosted on Sourceforge.net (SF), to examine these hypotheses more closely. Although our SF data lack some potentially important factors, we believe they have revealed important information about what makes projects succeed. We initially presented this paper at the FOSS4G 2008 conference held in Cape Town, South Africa [25]. The first part of this paper is taken directly from our earlier paper, and describes our literature research and the factors that may be important for collaborative success in open source commons.. The second part of the paper has been updated to reflect the latest results and conclusions from our empirical research.

2. Factors Thought to Influence Open Source Collaborations

We have reviewed a sizeable amount of theoretical and empirical literature in a variety of disciplines searching for factors thought to contribute to the success or abandonment of open source collaborations. We started with the traditional information systems development literature, but then moved to literature on distributed work and virtual teams, as well as literature on collective action and commons governance and management. Much of this latter work focuses on collaborations in natural resource commons or common property, but very recently scholars have begun studying “digital commons,” such as open access publishing, and open content collaboration [10]. In this section we provide an overview of the variables we have identified through this process. Following Ostrom [17], we organise them into physical, community and institutional attributes (Figure 1).

[1] For reasoning behind why we call these projects a commons, see our open access paper published in *Upgrade* [22a].



Physical Attributes
- Software requirements
- Modularity and granularity
- Product Utility*
- Competition
- Collaborative Infrastructure used
-Versioning system
- Bug tracking*
- Communication technologies*
Community Attributes
- User Involvement*
- Leadership*
- Social Capital
- Group Homogeneity/Heterogeneity
- Group size*
- Financing
- Marketing strategies
- Clear vision*
Open Source Commons Success or Abandonment
Institutional Attributes
Levels: Operational, Collective Choice, Constitutional*
Rule Types: (1) Position rules, (2)Boundary rules, (3) Choice rules, (4) Aggregation rules, (5) Information rules, (6) Payoff rules, (7) Scope rules.
Note: "*" denotes concepts that we could operationalize using the Sourceforge.net dataset

Figure 1. Factors Thought to Influence the Success or Abandonment of Open Source Collaborations (Source: The authors)

2.1 Physical Attributes of Open Source Commons

Physical attributes refer to the set of variables related to the software developed or to some of the technological infrastructure needed for team coordination. Our review identified the following variables that potentially affect the success or abandonment of open source commons: (1) software requirements, (2) modularity, (3) product utility, (4) competition, and (5) collaborative infrastructure.

Software requirements refer to the processes used to determine what the software will or should do. It is thought that projects with clearly defined visions will do better than ones without such visions. *Modularity* has to do with the design of the software, and whether it is easily broken down into separate, relatively standalone components. Within limits, a modular design is thought to make it easier for contributors to "carve off chunks" of the project that they intend to work on [30]. *Product utility* means that a project will be more successful if the software being produced is something that people want or need, or if it is more useful than other software products [30]. *Competition* refers to whether the project is unique in what it is trying to do, or whether there are other similar projects available. Significant competition would lead to potentially fewer available people or organizations wanting to join in to any particular project. Competition also captures the situation where a rival technology comes along that reduces people's interest in the product being developed. Finally, *collaborative infrastructure* describes the types of technologies used to help coordinate the collaborative team. There are a variety that may be used, including a code version control system, a bug tracking system, and a number of communication and documentation technologies (e.g., email lists, web-based forums, Internet Relay Chat, etc.). The particular configuration may be particularly important in reducing the time expended by team members. For instance, a norm emphasizing the use of a question and answer forum for help creates, at the same time, a searchable documentation database.

2.2 Community Attributes of Open Source Commons

This label captures variables related to the people who are developing the software, along with the financial and marketing aspects of the project. These include: (1) user involvement; (2) leadership; (3) social capital; (4) group homogeneity/heterogeneity; (5) group size; (6) project financing; and (7) marketing strategies.

User involvement is one of the long-standing variables known to influence the success or failure of traditional software development projects [6] as well as in open source settings [28, 29]. Similarly, the concept of *leadership* appears repeatedly in the literature and is thought to be a success or abandonment factor in both traditional face-to-face teams and virtual teams [26]. Components include leadership structure, how well the leader(s) are able to motivate the team, and how well goals or a "clear vision" are articulated [12]. In political science and economics, the degree of *social capital* – usually characterised as "trust" between community members – is often discussed when describing a "healthy" or vibrant community [19]. In other commons settings three factors contribute to the establishment and maintenance of trust: reciprocal relationships (e.g., I help you, you help me), repeated interactions [18], and regular face-to-face meetings [14].

Group heterogeneity is thought to influence the ability for a team to act collectively [21]. Varughese and Ostrom [27] sub-divide the concept into three categories: (1) socio-cultural, (2) interest, and (3) asset heterogeneity. Socio-cultural heterogeneity includes attributes such as ethnicity, religion, gender, caste, language, or other cultural distinctions. The presumption is that groups with diverse socio-cultural backgrounds will have more difficulties working together because of a lack of understanding and, potentially, a lack of trust. Interest heterogeneity captures the motivations of people who participate in a commons. Volunteers often participate in open source for different reasons than some paid programmers. It is an open question as to whether diverse or diverging interests in open source affect collaboration, although some literature suggests that tensions could arise when heterogeneous interests (such as volunteer and business) coincide [19, 27]. Lastly, asset heterogeneity captures the idea that some individuals may bring to a project capabilities or resources that others on the team might not have themselves. For example, concepts like wealth and political power are two types of assets found in some commons settings. Studies related to natural resource commons have found that heterogeneity in assets negatively impacts a group's ability to self-organise [11].

Group size is another variable that has long been thought to influence success or failure in natural resource commons as well as software development settings [24]. For years it has been thought that the larger the group the higher the coordination costs [16, 2]. Yet specifically in open source, “Linus’ Law” – “with more eyes, all bugs are shallow” [20] – suggests that larger groups are actually helpful. Others have found empirical evidence that suggests that the relationship between group size and success is complex, not direct, and probably not linear. For instance, changes in group size tend to simultaneously affect other variables, such as group homogeneity and leadership [4]. In short, group size has long been thought to be influential, but its effect on open source commons is unclear.

The last two community attribute variables are *project financing* and *marketing strategies*. Several open source researchers emphasise financing as a key variable for project success [30, 8]. The argument is that financing can ensure that someone is working on the project and provides some assurance that the project will move ahead. At the same time, funding from a particular source could lead to some tensions over future technical direction of the project in the case where there is a hybrid (e.g., volunteer and paid developer) team. Turning to marketing, surprisingly, there appears to be very little in the literature on this as a variable that affects open source success or abandonment. Yet there are indirect suggestions in the literature about the importance of getting the project known in the early days to gain a user community as well as more development support.

2.3 Institutional Attributes of Open Source Commons

The “institutional attribute” category in Figure 1 contains variables related to the governance and management systems used by the open source commons and the types of rules in place intended to guide the behavior of participants. Institutions are a key set of variables in natural resource commons used to protect the resource from overuse. However, it is only very recently that researchers are conceptualizing and investigating empirically institutional designs in open source settings [22, 15, 13, 23]. Part of the reason for this lack of attention may be that formal rules and procedures are thought to create disincentives for participating in open source [20], and in open source commons content version control systems replace the need for rules by protecting the software from accidental destruction through version control and rollback functionality. However, given the emerging involvement in open source development by firms, government agencies and nonprofit organizations, it is likely that institutional designs will increasingly be a factor in whether projects succeed or become abandoned.

To analyze open source institutional designs, we build specifically on the work of Elinor Ostrom [17] who organises institutions into Operational, Collective Choice and Constitutional levels. Operational norms and rules oversee the day-to-day activities in a project. Collective choice rules define how changes to operational level rules occur and who has the authority to make such changes. Constitutional level rules specify who is eligible to change Collective Choice rules and also define the procedures for making such changes. They also can be formalised rules that establish the boundaries or principles that the collaboration is grounded upon. The project’s open source licence is the obvious example of a Constitutional level element. Within each level are seven types of rules [17] (shown in the "Institutional Attributes" box in Figure 1), but because of space limitations and because Sourceforge.net data does not contain such information, we do not describe this more fully here.

3. An Empirical Analysis of SourceForge.net Projects

As most readers of this paper will know, Sourceforge.net (SF) is the largest hosting website for open source software projects. The FLOSSmole project [7] collects data from SF and makes it publicly available. Pre- and post-release data on projects are contained in this dataset. In earlier work, we divided the lifetime of open source software projects into an Initiation Stage and a Growth Stage [22, 22a]. The Initiation Stage is the period from the time the project begins until the time that it has its first public release of software. The Growth Stage is the time period after the first public release. Using FLOSSmole data, along with data we gathered from SF ourselves, we classified 107,747 projects as either successful or abandoned in the Initiation or Growth Stage, and then manually validated this classification to make certain it was accurate. Table 1 summarises our definitions. For more information, see [5]. This classification became our dependent variable. For our independent variables, we used six numerical and seven groups of categorical variables present in the SF data. Table 2 describes the six numerical variables and indicates some of the "subcategories" which comprise each of the seven "groups" of categorical variables. Each subcategory is an individual independent variable in our analysis. The reader may also refer back to Figure 1, where we indicated the variables present in our SF data with an asterisk. A quick glance at Figure 1 reveals that many potentially important variables are not included in the SF data set.

Table 1. Summarised Success and Abandonment Classes (for full detail, see [5])

Success, Initiation	Developers have produced a first release
Abandonment, Initiation	Developers have not produced a first release and >1 year since project registration
Success, Growth	Project has achieved three meaningful releases and the software has been downloaded more than 10 times.
Abandoned, Growth	Project appears to be abandoned (no activity) before producing three releases

Table 2. Selected Variables in Sourceforge.net Data

SF Variable	Description	Associated Theoretical Concept (Figure 1)
Developers	Total number of developers on the project	Group size – Community attribute
Tracker Reports	Total number of bug reports, feature requests, patches and support requests	Collaborative infrastructure – bug tracking system. Physical attribute. User Involvement – Community attribute. Group size – Community attribute

Page Visits	Total number of views of any of the project's SF website	Product utility – Physical attribute; Group size – Community attribute
Forum posts	Total number of Forum posts made to the project's public forums from 2005-10-06 through 2006-08-02	Collaborative infrastructure – Physical attribute; Group size – Community attribute
Downloads	Total number of downloads of the software package	Product utility – Physical attribute; Group size – Community attribute
Project Information Index	Total number of "subcategories" of the categorical variables (described below) that a project has selected to describe itself.	Product utility – Physical attributes
Intended Audience	Categorical variable describing the type of person the project targets (e.g., end users, advanced end users, business, computer professionals, other)	User Involvement – Community Attribute
Operating System	Categorical variable describing the operating system(s) the software will run on	Product utility, critical infrastructure – Physical attribute
Programming language	Categorical variable(s) describing the programming languages used	Product utility, preferred technologies – Physical Attribute
User Interface	Categorical variable describing how the software interfaces with the user (e.g., command line, GUI, etc.)	Product utility, preferred technologies – Physical Attribute
Database Environment	Categorical variable for the database used in the project's software (if relevant)	Product utility, preferred technologies – Physical Attribute
Project Topic	Group of 19 categorical variables consists of the topics that the SF website uses to classify the projects (e.g., education, games, security, printing, etc.)	Product utility, critical infrastructure – Physical attribute
Project License	Categorical variable(s) describing the type of open source license(s) used.	Constitutional rules – Institutional Attribute

3.1 Statistical Methods: Classification Tree Analysis.

Classification trees [3] are a method of dividing data into dependent variable groups based on independent (sometimes referred to as "predictor") variables. In our case, classification trees use our six numerical and seven groups of categorical independent variables to attempt to divide the data as accurately as possible into groups of successful or abandoned projects. To put it another way, classification trees attempt to use each and every variable to maximise the accuracy of predicting whether a project is successful or abandoned. We had several reasons for choosing classification trees over other forms of statistical analysis, such as logistic regression. First, classification trees can easily handle numerical as well as categorical data and model complex interactions [1]. Second, this technique is non-parametric, thus eliminating questions about the suitability of the data for parametric analysis. Finally, classification trees are fairly easily interpreted. Since the classification technique is computationally demanding, we were not able to simultaneously process all 107,747 projects in our database. Instead, we used random samples of the data that were large enough to produce consistent results.

3.2 Example of Classification Tree Results.

The classification tree shown in Figure 2 was created from a random sample of 1,000 projects in the Growth Stage, and the tree algorithm used all our independent variables as potential predictors of success or abandonment. The algorithm found that the most accurate way to discriminate projects that were successful in the Growth Stage ("SG" in the figure) from projects that had been abandoned in the Growth Stage ("AG") was to first divide the data based in the number of Page Visits. This first division, or first "node" of the tree, divided the projects according to whether they had greater than or less than 4,317 Page Visits. The data sample included 524 projects having less than 4,317 Page Visits (shown on the left hand side of the tree), and 85% of them were "correctly classified" (abbreviated "cc" in the tree) as AG projects. Projects with greater than or equal to 4,317 Page Visits were further divided (in the second node on the right hand side of the tree) as to whether they had less than four Tracker Reports. Figure 2 shows that 276 projects had less than 4 Tracker Reports and were classified as AG; however, only 56% of these projects were correctly classified. On the other hand, of the 200 projects that had four or more Tracker Reports, 80% were correctly classified as SG. Overall, this model correctly classified 76% of the projects. The Kappa statistic reveals that the model improved the classification by 42.5% over what would have resulted if the projects had been divided by chance. The AG row in the confusion matrix shows how many AG projects (601) were correctly classified as AG and how many AG projects (40) were incorrectly classified as SG. Similarly, the SG row in the matrix shows how many SG projects were correctly or incorrectly classified. We also constructed trees from random samples of projects in the Initiation Stage (not shown), and we will describe the results for both stages in the following "Findings" sections.

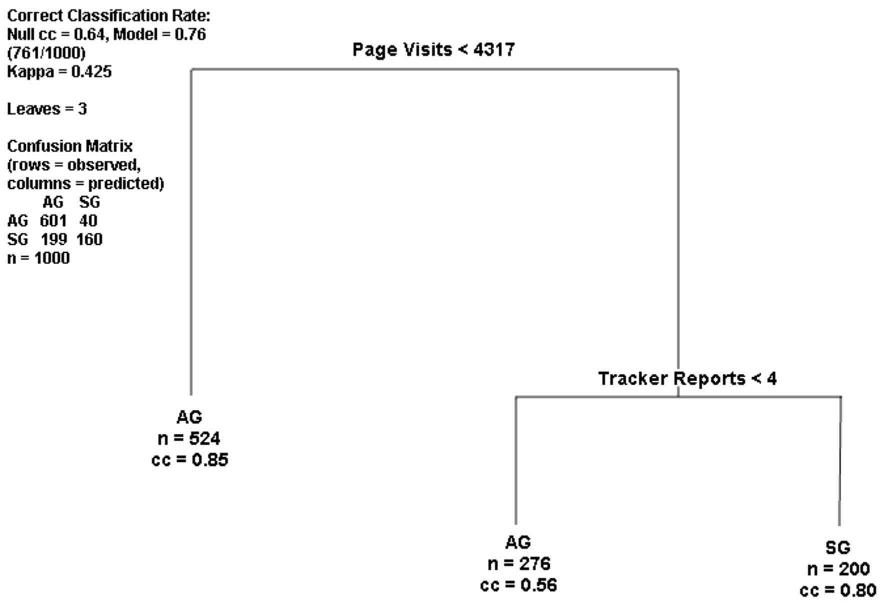


Figure 2. Example of Classification Tree Results Using 1000 Randomly Sampled SF Growth Stage Projects

3.3 Initiation Stage Findings

As we have suspected for some time, different factors discriminate successful from abandoned projects in the Initiation Stage as opposed to the Growth Stage. Recall that we defined the Initiation Stage as the time before a project has its first public release. Classification trees produced using random samples of Initiation Stage projects had similar accuracy to the tree shown in Figure 2, and revealed the Project Information Index (PII) to be the top splitting variable. The PII is the sum of the number of descriptive categories chosen by a project's developers to describe the project. For example, if a project description on SF included three different intended audiences, two different programming languages, a single project topic and no other descriptive categories, then the PII would have a value of six. We were able to determine, by examining projects at the time of their first release, that projects that have much higher than average PIIs in the Initiation Stage tend to become successful in Initiation, rather than becoming abandoned.

But why would the PII be the most important factor for separating successful from abandoned projects in the Initiation Stage? We hypothesised that there might be three explanations, based on factors that turned up in our literature review. First, a project that has a higher PII will often have higher "utility" because higher PII likely means that the project has a broader functionality. For example higher PII might reflect a broader intended audience, the ability to work with more programming languages, the ability to run on more operating systems, or other useful functionality related to the categorical variables. Second, having a "clear vision" for the project would likely result in a more accurate description of the project, and thus a higher PII. Finally, a higher PII may indicate more diligent and engaged leadership, simply because the "leader" takes the time to provide a detailed description of the project. We can make a reasonably good case that higher PII is a "cause" of success in Initiation because (1) higher PII occurs before a project becomes successful, (2) higher PII is highly correlated with success, as revealed by the classification tree results, and (3) we can envision mechanisms related to PII that could reasonably help a project to become successful. However, we cannot be sure whether higher utility, clear vision (i.e setting goals), leadership or any combination of these three factors are the key to success. We intend to investigate this finding further in upcoming survey research.

3.4 Growth Stage Findings

As described above, the top splitting variable in Figure 2 is Page Visits, which we have listed in Table 2 as being associated with the concept of utility. The second most important splitting variable is Tracker Reports, which we have hypothesised is related to user involvement and group size. We base this Tracker Reports hypothesis on the idea that a project with more users, or more involved users, seems more likely to have Tracker Reports, since Tracker Reports are comprised of bug reports and feature requests, among other things. We have also found, using statistical methods not described here, that the average number of *developers* on projects increases before a project becomes successful in the Growth Stage, and that higher developer counts are correlated with success. So it appears that useful software, and probably other factors, attract a larger user and developer community and that a significant community is a defining characteristic of successful Growth Stage projects. This finding may seem to be intuitively obvious, but other findings were possible because we specifically designed our definition of success to include projects that produce software that is only of interest to a few users. An example of important software that might be used by only a few users could be a bioinformatics software that studies a gene involved in a rare disease. These small, but successful, projects are probably among the 15% of the 524 projects, shown in the left hand side of the tree in Figure 2, that are misclassified as AG projects. It could have turned out that the majority of successful Growth Stage projects were small with little evidence of community, but this is not the case. In summary, we have found that the majority of successful Growth Stage projects have a substantial community of users and developers, and that building such a community is a characteristic of success in the majority of growth stage projects.

Another thing worth noting in Figure 2 is that none of our categorical variables appear in the tree as important predictors of success or abandonment in the Growth Stage. By querying our database and looking at the distribution of successful and abandoned projects across all our categorical variables, we found that there are many successful and abandoned projects, in both the Initiation and Growth Stages, in every categorical variable subcategory. In other words, open source software is a broad-based phenomenon that spans Intended Audiences, Operating Systems, Programming Languages and our other categorical variables, and thus, these characteristics are not very useful for discriminating successful from abandoned projects.

4. Conclusions

We started this paper by describing how understanding open source commons is important for understanding the significant social and economic changes that have occurred as a result of this phenomenon. In addition, understanding these commons could help in forming collaborations to solve critical global problems. We began our investigation by studying multidisciplinary literature to discover factors that might be important for the success or abandonment of these commons. We then linked these factors, and related theoretical concepts, to data available from SF. We described how we created and validated a dependent variable that classified 107,745 SF projects as successful or abandoned in the Initiation and Growth Stages. We then used classification trees, among other statistical techniques, to discover which of the factors, represented by various independent variables available in the SF data, were most able to discriminate successful from abandoned projects. We found that the potential utility of the software, having a clear vision or goals for the project, and diligent leadership were likely to be causes of success in the Initiation Stage. In the Growth Stage, building a community of users and developers around a software product that is useful to a fairly large number of people is associated with success in most cases. Finally, we noted that many potentially important variables are not represented in the SF data. Missing variables include things like: skill levels of developers, homogeneity/heterogeneity of the developer teams, software complexity, project financing, and institutional structures. We intend to continue our research to study these potentially important factors and to further investigate the findings discussed in this paper.

Acknowledgments

Support for this study was provided by grants from the U.S. National Science Foundation (NSFIIS 0447623 & 0630239). Any opinions, findings, conclusions, or recommendations expressed in this material are the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Breiman, L, Friedman, JH, Olshen, RA, and Stone CG, 1984, *Classification and Regression Trees,* Wadsworth International Group, Belmont, California.

[2] Brooks FP Jr. [1975] 1995, *The Mythical Man-Month: Essays on Software Engineering,* Anniversary Edition, Addison-Wesley, Reading, MA.

[3] De'ath, G & Fabricius, KE, 2000, 'Classification and regression trees: a powerful yet simple technique for ecological data analysis', *Ecology,* vol 81, no. 11, pp. 3178-3192.

[4] Deek, FP and McHugh JAM, 2008, *Open Source Technology and Policy*, Cambridge University Press, New York, NY.

[5] English, R & Schweik, CM, 2007, 'Identifying success and abandonment of free/libre and open source (FLOSS) commons: a preliminary classification of sourceforge.net projects', *Upgrade: The European Journal for the Informatics Professional*, vol VIII, no. 6, viewed 20 July 2008, <http://www.upgrade-cepis.com/issues/2007/6/upg8-6English_Schweik_v2.pdf>.

[6] Ewusi-Mensah, K, 2003, *Software development failures*, MIT Press, Cambridge, MA.

[7] FLOSSMole, 2008, viewed July 22, 2008, <http://sourceforge.net/projects/ossmole/>.

[8] Fogel K, 2007, *Producing Open Source Software: How to Run a Successful Free Software Project*, O'Reilly Media, Sebastopol, CA. Viewed 13 July 2008, <http://producingoss.com/>.

[9] Ghosh, 2006. "Economic Impact of Open Source Software on Innovation and the Competitiveness of the Information and Communication Technologies (ICT) Sector in the EU. Final Report. Available at http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf. Accessed March 9, 2008.

[10] Hess, C, and Ostrom, E (eds.), 2007, *Understanding Knowledge as a Commons: From Theory to Practice*, Cambridge, MA, MIT Press.

[11] Issac, MR & Walker J, 1998, 'Group size effects in public goods provision: the voluntary contribution mechanism', *Quarterly Journal of Economics*, vol 53, pp. 179-200.

[12] Katzenbach, J & Smith D 1993, 'The discipline of teams', *Harvard Business Review*, vol 71, pp. 111-120.

[13] Markus ML, 2007, "The governance of free/open source software projects: monolithic, multidimensional,or configurational?", Journal of Management and Governance, vol. 11, no. 2, pp. 151-163)

[14] Nardi, BA & Wittaker S 2002, 'The place of face to face communication in distributed work', in P Hinds, and S Kiesler (eds.), *Distributed Work*, MIT Press, Cambridge, MA.

[15] O'Mahony, S & Ferraro F 2007, 'Emergence of governance in an open source community', *Academy of Management Journal*, vol. 50, no. 5, pp. 1079-1106.

[16] Olson M, 1965, *The Logic of Collective Action*. Harvard University Press, Cambridge, MA.

[17] Ostrom E, 2005, *Understanding Institutional Diversity*, Princeton University Press, Princeton, N.J.

[18] Ostrom E, Burger J, Field CB, Norgaard RB, & Policansky D 1999, 'Revisiting the commons: local lessons, global challenges', *Science*, vol. 284, pp. 278-282.

[19] Putnam RD 2007, 'E pluribus unum: diversity and community in the twenty-first century. The 2006 Johan Skytte prize lecture', *Scandinavian Political Studies*, vol. 30, no. 2, pp. 137-174.

[20] Raymond E, 2001, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly, Sebastopol, CA.

[21] Sandler T, 2004, *Global Collective Action*. Cambridge University Press, Cambridge MA.

[22] Schweik CM.& Semenov A 2003, 'The institutional design of 'open source' programming: implications for addressing complex public policy and management problems', *First Monday*, vol. 8, no. 1, viewed July 23, 2008, <http://www.firstmonday.org/issues/issue8_1/schweik/>.

[22a] Schweik, C.M. 2005. "An Institutional Analysis Approach to Studying Libre Software 'Commons'". *Upgrade: The European Journal for the Informatics Professional.* June. pp 17-27. Available at http://www.upgrade-cepis.org/issues/2005/3/up6-3Schweik.pdf. (Spanish version available at http://www.ati.es/novatica/).

[23] Schweik CM & English R 2007, 'Conceptualizing the institutional designs of free/libre and open source software projects', *First Monday*, vol. 12, no. 2, viewed 23 July, 2008, <http://www.firstmonday.org/issues/issue12_2/schweik/index.html>.

[24] Schweik CM, English R, Kitsing, M. & Haire, S 2008, 'Brooks' versus Linus' law: an empirical test of open source projects', in SA Chun, M Janssen, and R Gil Garcia (eds.), *The Proceedings of 9th International Digital Government Research Conference*, Montreal, Canada.

[25]Schweik, C.M., English, R. and Haire, S. 2008. "Factors Leading to Success or Abandonment of Open Source Commons: An Empirical Analysis of Sourceforge.net Projects." Paper presented in the Academic Paper track of the 2008 Free and Open Source for Geospatial Conference, Cape Town, South Africa. September 29 October 4, 2008. Abstract available at http://conference.osgeo.org/index.php/foss4g/2008/paper/view/135

[26] Tyran KL, Tyran, CK, & Shepherd M 2003, 'Exploring emerging leadership in virtual teams', In CB Gibson, and SG Cohen (eds.), *Virtual Teams that Work: Creating Conditions for Virtual Team Effectiveness*. Jossey-Bass, San Francisco, CA.

[27] Varughese G & Ostrom E 2001, 'The contested role of heterogeneity in collective action: some evidence from community forestry in Nepal', *World Development*, vol. 29, no. 5, pp. 747-765.

[28] von Hippel E & von Krogh G 2003, 'Open source software and the 'private-collective' innovation model: issues for organization science', *Organization Science*, vol. 14, no. 2, pp. 209-223.

[29] von Hippel E, 2005, *Democratizing Innovation*. MIT Press, Cambridge, MA.

[30] Weinstock, CB & Hissam, SA 2005, 'Making lightning strike twice', In J Feller, B Fitzgerald, SA Hissam and K R Lakhani (eds.), *Perspectives on Free and Open Source Software*. The MIT Press, Cambridge, MA.