International Journal of the Commons
Vol. 7, no 2 August 2013, pp. 234–254
Publisher: Igitur publishing
URL:http://www.thecommonsjournal.org
URN:NBN:NL:UI:10-1-114926
Copyright: content is licensed under a Creative Commons Attribution 3.0 License
ISSN: 1875-0281

Preliminary steps toward a general theory of Internet-based collective-action in digital information commons: Findings from a study of open source software projects

Charles M. Schweik
University of Massachusetts, Amherst, USA
cschweik@pubpol.umass.edu

Robert English
University of Massachusetts, Amherst, USA
bobengl@gmail.com

Abstract: This paper presents some of the findings from a 5-year empirical study of FOSS (free/libre and open source software) commons, completed in 2011. FOSS projects are Internet-based common property regimes where the project source code is developed over the Internet. The resulting software is generally distributed with a license that provides users with the freedoms to access, use, read, modify and redistribute the software. In this study we used three different and very large datasets (approximately 107,000; 174,000 and 1400 cases, respectively) with information on FOSS projects residing in Sourceforge.net, one of the largest, if not the largest, FOSS repository in the world. We employ various quantitative methods to uncover factors that lead some FOSS projects to ongoing collaborative success, while others become abandoned. After presenting some of our study's results, we articulate the collaborative "story" of FOSS that emerged. We close the paper by discussing some key findings that can contribute to a general theory of Internet-based collective-action and FOSS-like forms of digital online commons.

Keywords: Collaborative success and abandonment, common property regime, digital information commons, free/libre software, open source software

Acknowledgments: Support for this work was provided by a grant from the U.S. National Science Foundation (NSFIIS 0447623). The findings, recommendations and opinions expressed are those of the authors and do not necessarily reflect the views of the funding agency. We also acknowledge co-funding for this publication from the European Commissions' 7th Framework Programme, under contract GENCOMMONS (European Research Council, grant agreement 284). We are especially grateful to post-doctoral fellows Sandra Haire and Meng-Shiou Shieh who provided statistical support. We also gratefully acknowledge the comments received at the presentation of an earlier draft of this paper at the 1st Global Thematic IASC Conference on the Knowledge Commons [Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, from September 12th to 14th, 2012.] and of three anonymous reviewers. Of course, any mistakes are our responsibility alone.

1. Introduction

To emphasize the importance of digital information as a commons, let us start with a question for readers' reflection: What allowed people to construct websites so rapidly and exponentially in the early years (1994–1999)?

We agree with publisher Tim O'Reilly (2003) on the answer: the early web browsers like Mosaic, Netscape Navigator and Internet Explorer all provided a "View Source" menu item, allowing web surfers to read the HTML logic of the webpage they liked, copy it, and then create new derivative works based on it. Moreover, we argue further that this period could be considered the most successful distance-learning phenomenon in human history, driven by the fact that the code was readable and open. Webpage growth from 1994 to 2000 is a global-scale existence proof of the potential innovative power of open access, digital information commons on the Internet.

Fast-forward to 2012 and scale down to the individual. Recently we witnessed another example of this innovation phenomenon based on openness and learning in a place we did not expect it – at home, where co-author Schweik was interacting with his 11-year-old son Max as he used the family computer. Max is just starting to learn a programming language called "Scratch" created at the MIT media lab for children. Max had developed an initial program that displayed an object that moved across the screen. But he became stuck on the problem of how to make it "loop" and reappear on the left screen edge when it went off the right edge side of the screen. Without asking for any help, he went to the MIT Scratch website, viewed other Scratch animations published by other kids, and found one that had the "edge-looping" behavior he envisioned. He then viewed and deciphered the source code (since the MIT site makes the code available for others to read), he interpreted that code, and figured out how to implement similar screen looping logic in his animation. This 11-year-old's activity was, at the individual level, an incredible demonstration of the power of open source and open access to digital

information on the Internet. Moreover, it is quite likely that this open learning phenomenon is occurring not only in software, but in other knowledge sharing, digital information situations.

In the cases above, the web page developer and Max are consumers and learners of open code who then use that knowledge and implement it for their own needs – what Eric von Hippel (2005) calls "user-centered innovation". Indeed, open knowledge commons can lead to learning and new innovation, but it could be argued that this innovation can be accelerated if harnessed in collaborative, co-production situations.

In the area of computer software development this kind of co-production has existed since the beginning of the computing era (1950s, 1960s), when the free sharing of readable software code and collaboration on new versions were the norm. It was only in the late 1960s and 1970s that software became viewed as a proprietary commodity by software development firms (Drahos and Braithwaite 2002; O'Reilly 2003). This is an early example of what Boyle (2003) refers to as the "Second Enclosure Movement" where "… things that were formally thought of as either common property or uncommodifiable are being covered with new, or newly extended, property rights" (p. 37). A decade or more later, this enclosure of software logic led Richard Stallman at MIT to come up with his brilliant push-back using copyright law; an approach he coined "copyleft" (Stallman 1999). Stallman created the General Public License (GPL) that gave the user the freedoms to access, use, read, modify and redistribute his GNU operating system software. Other open source software licenses followed shortly thereafter and differ from free/libre licensed software in some respects,[1] but for our purposes, we will treat the labels free/libre and open source software as examples of the same general class of software and will refer to them collectively as "FOSS". Since Stallman's innovation in the 1980s, FOSS has continued to grow and is now widely and globally deployed.

As a result of the open-source phenomenon, perhaps more than any other category of Internet user, computer programmers and FOSS programmers in particular have significant experience in online collaboration in the context of digital information commons. The study of FOSS commons is important not only because of its potential for learning and innovation, but because uncovering underlying collaborative principles could enhance not only software co-production but also collaboration in other digital commons contexts.

Scholars have recognized the importance of FOSS collaboration. Over the last decade, a sizable amount of research has been conducted on FOSS (Aksulu and Wade 2010). After their review of the literature, Crowston and colleagues (2012) conclude that the body of research is biased toward well-established, successful FOSS cases and largely ignores unsuccessful projects; and focuses less attention

[1] (There are more subtle differences between "free/libre" software and "open source" software that are not important for our discussion here. For more information see http://opensource.org/faq#copyleft or http://open source.org more generally.)

on projects in "initial or transition phases" or on the evolutionary nature of FOSS. Moreover, we would add that to date there are no large-scale empirical studies focusing on the socio-technical factors that lead FOSS projects toward ongoing collaborative success or abandonment and more accurately portrays the active population of FOSS projects.

In the original call for papers for the International Association for the Study of the Commons' 1st Thematic Conference on "the Knowledge Commons", for which this paper was initially written, it stated: "The motivating questions for this conference is how best to devise and diffuse institutional and organizational models that would maximize social benefits and returns from the knowledge commons, by promoting broad access to and reuse of research resources, rather than restricting it; and how this can be done while preserving reputational benefits and essential ownership rights, as well as transparent and shared quality standards". Given that FOSS programmers have been collaborating over the net on common property knowledge commons for multiple decades, understanding how they do this in a carefully designed, systematic way, is vital for answering this question.

In this paper, we summarize findings from a 5-year empirical study of FOSS commons that attempted to close some of these gaps in our knowledge (Schweik and English 2012). The central research question we asked is: *What factors lead some FOSS commons to ongoing collaborative success and others to abandonment*? In the work we will describe next, we analyze two large datasets with information about a large number of FOSS projects, one representing approximately 107,000 in 2006, and another representing roughly 174,000 projects in 2009. These projects are from the FOSS hosting site Sourceforge.net. We complement the 2009 dataset with our own random-sample survey of over 1400 software Sourceforge developers. The project tested over 40 hypotheses and research questions related to success and abandonment of FOSS commons, including an analysis of project governance. In this paper, we briefly describe methods used and summarize some of our key findings. We conclude with some theoretical reflections that move toward a general theory of FOSS-like online collective-action.

2. Foundational theory

2.1. FOSS as common property peer-production

Let us begin with a clarification about FOSS projects as a form of digital information commons. Benkler (2006, 61–62) introduced the phrase "commons-based peer-production" to describe circumstances where no centralization exists and hierarchically-based assignments do not occur. Individual participants select their tasks to work on. While these practices can occur in various digital information settings, Benkler (2006, 63) puts forth FOSS as the "quintessential instance" of commons-based peer production. It is here that we would add further clarity for a readership specifically interested in commons issues. Individual

FOSS projects could be more accurately referred to as "peer-production common property" (Schweik 2005). In FOSS projects, explicit property rights exist that are supported by the associated FOSS license. In any given project certain developers have authority and access to the version control system that is used to manage the current and future releases of the code, and these developers have significant control over what enhancements get put into the next release. FOSS projects, viewed as common property peer-production, raise questions about social relations between developers on the team, as well as questions about management and governance of these collaborations.

2.2. The guiding Institutional Analysis and Development Framework

Initially, our investigation required research on: (1) how to conceptualize and measure the dependent variable "collaborative success or abandonment", and (2) what various factors might affect collaborative success or abandonment in Internet-based common property settings. We used the Institutional Analysis and Development (IAD) Framework – a framework well known to commons scholars – to help guide us as we combed through relevant theoretical and empirical literature (Figure 1).

Specifically, we reviewed theoretical and empirical research related to software engineering and information systems development, distributed work and virtual teams, social movements, collective action, and commons governance and management, much of which describes natural resource commons settings, but some of the more recent research focuses on the "new digital commons" (Hess and Ostrom 2007; van Laerhoven and Ostrom 2007; Bollier 2008). As we reviewed these literature, we developed theories about what factors might influence whether a FOSS project continues to be worked on or becomes abandoned, and we translated these theories into over 40 testable hypotheses, or sub-research questions where no *a priori* hypothesis could be articulated.

The left side of Figure 1 provides a list of some of the factors we investigated, organized into "categories" of project attributes: technological, community and institutional. Technological attributes refer to aspects of the software being developed, or the collaborative technologies used to coordinate work. Community attributes captures the people doing the development, or the user community using the software. We also lump financial and marketing efforts into this sub-category of independent variables. Institutional attributes describe generally the governance system that oversees a project. As institutional analysts of the commons know, this captures various rules-in-use which may be either formally articulated (e.g. put in writing) or could be simple unwritten social norms that developers follow in their day-to-day activities. In the FOSS context, one important constitutional level set of rules is the software license used, such as the Free Software Foundation's General Public License (GPL) and its variants, or a wide variety of non-GPL compliant licenses (see OSI 2012 for more detail).

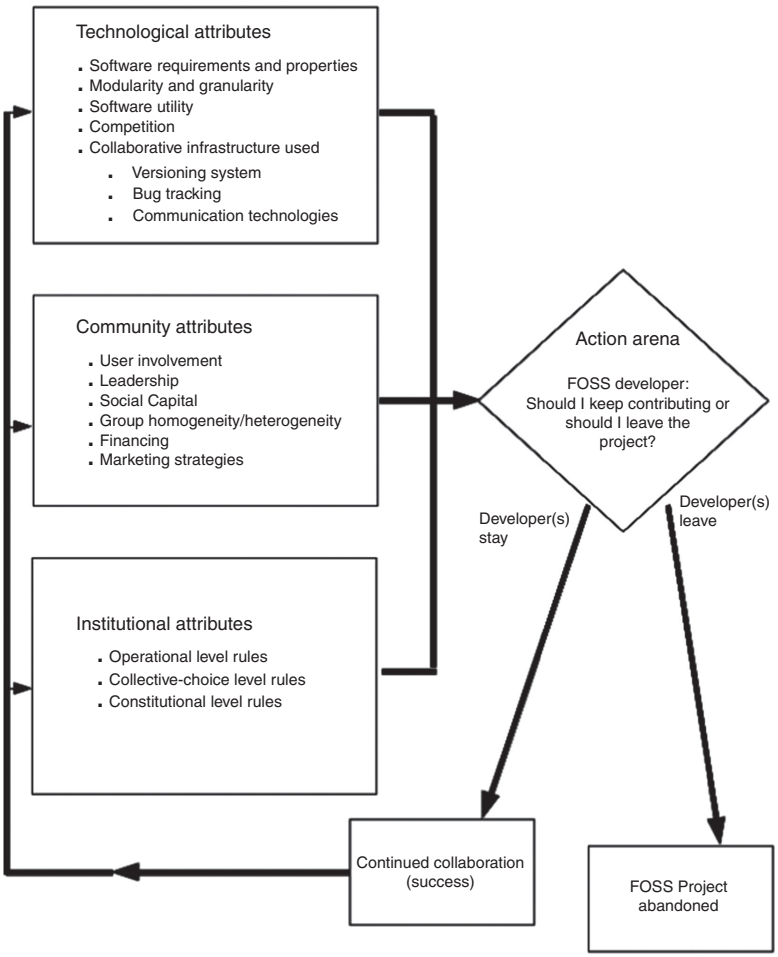

Figure 1: A simplified Institutional Analysis and Development Framework to support analysis of FOSS common property regimes.

On the right side of the IAD framework in Figure 1, we capture the "action arena" (Ostrom 2005) that depicts one or more FOSS developers in any one point in time reflecting on whether he or she should either continue to contribute to the project or, alternatively, leave the project. Historically, FOSS was about volunteer programmers donating their time and intellectual property and the motivations for this behavior has been thoroughly researched (see, for example, David and Shapiro 2008). Increasingly, however, some developers are paid to contribute by their employer (Riehle 2007; Schweik and English 2012, Chapter 2). Regardless of whether the developer is volunteer or paid, we assume from time to time there is self reflection on whether the time spent contributing to the project is deemed

valuable or not to the individual developer or his or her employer paying him or her for contributing to the project. One trajectory over time is that all developers "jump ship" and the FOSS project ultimately becomes abandoned (right bottom, Figure 1). In instances where one or more developers decide to stay and continue software maintenance or ongoing development (the "continued collaboration" box at the bottom right of Figure 1), the project continues to be worked on an a new period in the dynamic system begins, feeding back to the independent variables where attributes over time change.

2.3. FOSS development trajectories

Our literature review also involved research on ways to measure success of FOSS projects, but we ultimately settled on the concept of collaborative success and project abandonment because of our interest in explaining collective action in this context. But measuring collaborative success and abandonment in FOSS is tricky, in part because of the longitudinal nature of these kinds of collaborations.

Figure 2 graphically describes common collaborative success and abandonment trajectories in FOSS settings, and also highlights a key temporal juncture: the first public release of the software. As we considered collaborative processes, we ultimately hypothesized that the factors that influence collective action of FOSS developers were likely different in the time prior to a first software release compared to the time after the first public release. For example, much of the FOSS literature discusses end-user involvement and interaction (Dafermos

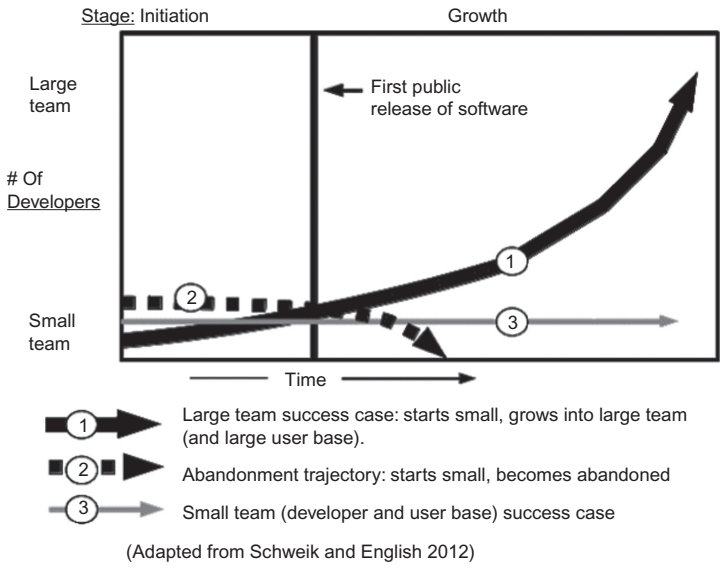

Figure 2: Key stages and trajectories in FOSS development.

2005; Fogel 2006), but that kind of involvement, by definition, would only occur after a first public release of the code.

Historically, the poster child of FOSS success has been the Linux operating system. In Figure 2, the dark bold Arrow 1 represents this kind of extreme collaborative success story. Linux started out as a 1-developer project in 1991, and over time, gained many developers (it now has perhaps 100s) and continues to release new and enhanced versions. The dotted line Arrow 2 in Figure 2 depicts the alternative trajectory. A project starts out with a small team, and over time – either before or after a first release – eventually loses the developers it has and becomes abandoned. However, the successful collaboration trajectory that tends to be forgotten is depicted by the thinner grey Arrow 3 in Figure 2. In this scenario, one or a small group of developers start the project and, over time, the size of the development team remains fairly stable but this team continues to work on the project. The example we like to use in this context is a small team developing FOSS in an area such as bioinformatics. In these kinds of instances, there is a small community of people who have the skills to develop such software, and there is probably a relatively small body of possible users (e.g. researchers in biology). But for our purposes, the trajectory of Arrow 3 is just as much a collaborative success story as the trajectory of Arrow 1. Our collaborative success dependent variable (described more fully below in Methods) will need to be able to handle both kinds of success stories.

3. Methods

Let us turn to a discussion of our methods used to investigate collaborative success and abandonment in FOSS commons. Given that we want to provide readers with more information on the results of the work and provide a discussion of their implications, we will only summarize key methods deployed here. Complete detail on methods (and results) is presented in Schweik and English (2012).

Our empirical work began in 2005–2006, and at the time we investigated ways to build a large database of FOSS projects for quantitative analysis that was potentially representative of the (unknown) population of FOSS projects. At the time, the dominant location on the web to find FOSS was the hosting site Sourceforge.net (SF). It was, and arguably still is the largest free/libre or open source software project-hosting site on the Internet (Deek and McHugh 2007, 152).[2]

It turned out that two different research groups – FLOSSmole (Howison et al. 2006) and the Sourceforge Research Data Archive (SRDA, van Antwerp and Madey 2008) were taking static "snapshots" of SF hosting data and making them available for researchers to use. Over the course of our project, we ended up using a FLOSSmole dataset representing 107,747 SF hosted projects in the year

[2] Although there are other hosting platforms that are now rivaling SF such as github (https://github.com/).

2006, and later obtained an SRDA SF dataset of 174,333 projects representing activity in 2009.[3] While these datasets provided an excellent foundation and covered many of the technological attributes of projects, they were missing data on community and institutional aspects (recall Figure 1). Consequently, with the help and support of the SF organization, we conducted a stratified random sample of 50,000 (!) projects from the 2009 database and invited project administrators to take an online survey of approximately 45 questions (one survey for Initiation stage projects, and one for Growth stage projects) designed to fill in the missing concepts. We estimated a sample of 50,000 was needed to ensure we would receive enough responses from abandoned projects. In the end, we received 1403 usable survey responses (683 surveys for Initiation Stage projects and 720 surveys for Growth Stage projects) from SF developers in 2009, which we then merged with our 2009 SF project metadata to create a database where all independent variables we wanted to investigate were measured in one, or sometimes in multiple ways.

With our datasets established, we turned to the issue of defining and operationalizing our dependent variable – FOSS project collaborative success or abandonment. We combined "use" and "popularity" measures (Crowston et al. 2003; Weiss 2005) with measures of project life and death (Robles-Martinez et al. 2003). Because we hypothesized that there might be different influential factors early in the project compared to later in the project, we applied different criteria for the "Initiation" (pre-first release) and "Growth" (post-first release) stages depicted in Figure 2. Table 1 provides definitions of these classes as well as classification results for the 2006 and 2009 datasets. These definitions meet the criteria we described earlier related to trajectories depicted in Figure 2; they capture success in instances where there are large or small development teams and user communities. Given the importance of this measure, we spent over a year figuring out measures of these concepts using SF project information, we classified the 2006 dataset based on our operationalization scheme, and we validated the classification by randomly sampling 300 projects and manually reviewing their project pages to make sure they indeed were correctly classified. We published a paper on this process (English and Schweik 2007) and Wiggins and Crowston (2010) independently verified our results.

With our dependent and independent variable concepts defined and operationalized, we turned to quantitative analysis. We approached this in three stages.

First, in 2007, we used our 2006 classification data and the 2006 FLOSSMole dataset to develop initial classification and regression tree-based models of FOSS

[3] As we contemplated these datasets, we were aware that some of the projects on SF are ones that were never meant to be ongoing projects nor had a goal of developing a community of users (e.g. computer science students using SF to host a programming project for a class, for example). But we expected this kind of "noise" in the SF data to fall out in statistical analysis given the number of projects SF hosts.

Table 1: FOSS collaborative success and abandonment classes, definitions and results for 2006 and 2009 SF datasets (For operationalization details, see Schweik and English 2012 Chapter 7)

Class	Definition	2006 dataset results (# of projects/%s)	2009 dataset results (# of projects/%s)
SI: Success in initiation	Developers have produced a first release	See*	See*
AI: Abandonment in initiation	No First release produced, and the project appears to be abandoned	37,320 (35)	67,126 (39)
SG: Success in growth	Project has achieved three meaningful releases of the software and the software is deemed useful for at least a few users	15,782 (15)	24,899 (14)
AG: Abandoned in growth	Project appears to be abandoned before producing three releases of a useful product, or has produced three or more releases in <6 months and is abandoned	30,592 (28)	53,450 (31)
II: Indeterminate in initiation	Project has yet to reveal a first public release but shows significant developer activity	13,342 (12)	16,806 (10)
IG: Indeterminate in growth	Project has not yet produced three releases but shows development activity, or has produced three releases or more in <6 months and shows development activity	10,711 (10)	12,052 (7)
Total projects		107,747	174,333

*Successful initiation (SI) numbers are not listed because these successes are growth stage projects. Including SI would double-count.

success and abandonment for Initiation Stage projects and for Growth Stage projects.

Second, after acquiring the 2009 SRDA dataset, in 2009 we administered the SF Developer Survey described earlier. We then used contingency tables to systematically analyze each survey question (and hence, almost all 40+ hypotheses).

Third, using the knowledge gained from the two previous steps, we once again used classification and regression tree modeling and logistic regression to analyze the 2009 SF survey and SRDA data combined in an effort to create multivariate models of success and abandonment for both the Initiation and Growth Stages capturing the concepts in Figure 1. In the section that follows, we summarize some of the main results of each of these analytic stages. Readers interested in more detail are encouraged to see Schweik and English (2012) Chapters 8, 10, 11 and 12.

4. Selected results: analysis of the 2006 FLOSSmole dataset

Our initial analysis of the FLOSSmole dataset of 2006 SF projects led to several discoveries:

1. The vast majority of FOSS projects in SF are small development teams (1–3 people) with a long tail where a few projects have more than 20 developers.

   This finding is not new; Krishnamurthy (2002) first reported it. But given that much of the FOSS literature continues to analyze large development team, high profile FOSS projects, we think this is an important point to reiterate.
2. In both stages, projects (a) with a clearly defined vision; (b) building software with clear utility; and (c) possessing leaders who lead by doing (e.g. putting the time in) and also are good at articulating project goals tend to be more successful than projects that do not possess these characteristics.
3. While important explanatory variables were similar between the Initiation and Growth stage models, their influence differed between stages.

   For example, the measures for "leadership by doing" and "well-articulated goals" were more important in the Initiation stage projects compared to the Growth stage projects. These results support our earlier contention that factors contributing to success are different in the earlier stage of the project compared to the later stage of the project.
4. Successful Growth Stage projects tend to have slightly larger development teams compared to abandoned Growth Stage projects, and we have strong statistical support for this.

   This finding led us to include two questions our SF developer survey regarding where these new developers came from, geographically. We will revisit this point later.
5. Growth Stage projects with higher measures user community interest (e.g. higher numbers of web-page visits, software downloads, bug reports and forum posts) are more often classified as successful.

   This finding lends support to the conventional wisdom of FOSS where user communities are involved and interact with the development team.

5. Selected results: analysis of our SF developer survey

With the analysis of the 2006 SF data completed, we then designed and implemented in 2009 the online survey to SF developers. Our ultimate goal was to build a more complete multivariate model of FOSS success and abandonment, which is why, at the same time, we collected a 2009 static time slice of SF projects from the SRDA program at the University of Notre Dame (van Antwerp and Madey 2008). But based on our theoretical and literature work, we had over 40 hypotheses to test about individual factors, so before we undertook multivariate modeling, we wanted to conduct univariate analysis first, examining each hypothesized factor individually, using contingency tables. In this section, we report some of the most important and interesting findings of this phase of our analysis, highlighting a few of the forty-some factors we identified in our literature and theoretical work.

1. Our 2009 survey results support our SF 2006 findings: a clearly defined vision, the utility of the software and project leadership attributes are associated with success in both stages.

   Six different questions in our Initiation and Growth surveys measured concepts related to the project having a clearly defined vision. Contingency tables revealed highly statistically significant results to support the hypothesis that projects with well-defined goals and plans are more successful than projects reporting less well-defined goals and plans. This is true for both Initiation and Growth stage projects. Contingency table analysis of questions related to product utility and aspects of team leadership revealed similar results.
2. Fine-scaled task granularity is associated with success in the Growth Stage.

   In his book *The Wealth of Networks* (2006), Yochai Benkler discusses the concept of task granularity. The idea is that people – especially people volunteering their time – will be more apt to participate if there are small fine-scaled tasks they can sign up to work on, rather than taking on larger tasks requiring more time and effort. In our survey we asked whether projects had explicitly established fine-scale tasks for some people to do. We found highly significant statistical support that suggests that SG projects have higher numbers of fine-scaled tasks in place for potential contributors compared to AG projects.
3. Financial backing is associated with Growth Stage success.

   Our survey results suggest that financial support for a project is more important in the Growth Stage than the Initiation Stage. Our Initiation survey contingency table reveals weak support for the idea that financial support helps a project get to a first release, but responses from the Growth Stage project survey reveals highly statistically significant differences between SG and AG projects. SG projects are more often associated with financial backing of some sort.
4. Sociocultural Heterogeneity is not a barrier to success.

   The results from the 2006 data analysis suggesting SG teams grow, at least slightly, led us to investigate this phenomenon more in our developer survey. We included a question about the geographic location of new developers that joined the project. In our results, we found that 99 out of the 190 multi-developer projects in the SG class or about 52%, had collaborations across continents – North America and Europe. This indicates that sociocultural heterogeneity is not a barrier to success. We will return to this point in the discussion section that follows.
5. The majority of FOSS projects in SF follow a "benevolent dictator" model of governance. In addition, we have slight statistical evidence that suggests larger SG projects have slightly more formalized systems of governance than their smaller sized counterparts.

Elinor Ostrom's (1990) seminal work "Governing the Commons" listed design principles found in long enduring natural resource commons settings. One of these design principles suggested that decision-making was democratic; that is participants in the commons have a say over its direction. While this is likely true to a certain degree in FOSS commons, what became apparent from our survey work is the vast majority of both abandoned and successful FOSS projects have a benevolent dictator-type governance structure where a leader is authorized and makes major directional decisions for the project. This result is very likely a result of the skewed nature of our large dataset toward small-team development projects.

Related to the second point above, the literature that exists in FOSS on governance suggests that programmers "just want to work" and resist formalized rules meant to guide or control behavior. Eric Raymond, a famous FOSS proponent once wrote (2001, 127, emphasis added): "The real free-rider problem in OSS software are more a function of *friction costs* in submitting patches [code fixes] than anything else… the number of contributors (and, at second order, the success of) projects is strongly and inversely correlated with *the number of hoops* each project makes a contributing user go through." In our data analysis, we find statistically significant evidence supporting the argument that as SG projects get larger, they move away from working under systems of social norms and more toward formalized systems of governance.

6. A number of factors we thought might help distinguish between successful and abandoned FOSS projects were found to have no statistical importance.

   In the Initiation and Growth Stage survey data, these include: modularity (a vast majority of both SI and AI projects report being modular in design); project complexity (found both in AI and SI projects); developer motivations [similar motivations were found in both AI and SI projects, including von Hippel's (2005) user-centered need – more on this in the discussion section; trust among developers; socio-cultural, motivational or asset heterogeneity across the development team].

6. Selected results: multivariate models based on our SF developer survey combined with the 2009 SF dataset from SRDA

Our third analytic effort returned us to multivariate modeling using a dataset constructed by using the combined 2009 SRDA data and our Initiation and Growth Stage developer survey data, respectively. Once again, for each stage, we used classification and regression trees, as well as logistic regression.

Our Initiation Stage multivariate model had only moderate ability to distinguish between success and abandonment. It correctly classified 488 out of the 683 projects in the dataset (a 71% correct classification) with about a

30% improvement over chance.[4] But what is quite interesting about the results is that while this analysis uses completely separate data from our earlier 2006 data analysis, our results closely mimic what we reported earlier: key factors that distinguish between successful and abandoned projects in the Initiation Stage are leading through doing and hard work, as well as articulating clearly project goals and a vision of the future.

The Growth Stage multivariate model, however, did quite well in differentiating between successful and abandoned projects. Here, we correctly classified 448 out of 500 projects – a 90% correct classification rate, and a 74% improvement over chance. The most prominent variables in this model were (in descending order by importance): downloads, bug tracker reports, leadership, and community.

We believe the downloads variable – a count of the total number of downloads of the software recorded at the time our SF database was acquired – captures a measure of the size of the external user community. In our data, the number of downloads is an important variable helping to distinguish SG from AG projects, suggesting that a larger user community is also associated with success in the Growth Stage. In Schweik and English (2012) we provide strong statistical evidence suggesting that both the size of the development team and the size of the user community are not merely correlated with success, but are causal factors for success.

Bug tracking reports captures, for each project, the number of errors or new feature requests, often posted by the project's user community. This variable, like downloads, was high in the results of variables that help distinguish between SG and AG projects. Like downloads, this variable provides another measure of user community activity and provides evidence that growing the user community is a causal factor in success.

The leadership variable we used in this model captures the same meaning as it did earlier in our Initiation Stage analysis. Its strong presence in our model suggests that elements of leadership, including hard work, planning, articulation of goals and other aspects of project management are highly correlated with success in the Growth Stage. We have strong evidence that leadership is a causal factor for success in the Initiation Stage and therefore, the Growth Stage.

Lastly, a variable we call "community" – an index combining a number of survey questions that get at the time the respondent spent helping users, the number of nonfinancial contributions made by people not listed as formal project developers, and questions related to other contributions made by these non-project contributors – was found to also have substantial ability to help discriminate between SG and AG projects. Again, this suggests an important role being played by people not recognized as part of the formal development team in FOSS projects.

[4] See Schweik and English (2012, 258–266) for a full explanation of the model results we are describing. Also, for interested readers, we have made all our data and statistical scripts available at http://www.umass.edu/opensource/schweik/supplementary.html.

7. Discussion: the FOSS story that has emerged

Our results show that important factors associated with success and abandonment differ between the two longitudinal stages. Moreover, the 5 years of work that we have only been able to summarize here produced fairly parsimonious models of success and abandonment for these two stages – especially for the Growth or post-first release Stage – that have fairly strong explanatory power. This work provides empirical evidence based on large datasets that describes how FOSS collaboration works. So what have we learned?

Importantly, we learned from our developer survey that Eric von Hippel's (2005) concept user-centered need is a major motivator for FOSS developers, regardless of whether their projects achieve collaborative success or not. Across the board, a vast majority of our respondents reported that they either initiated or participated in a FOSS development project because they, or the organization they work for, are *users* of the software they work on.

That being said, FOSS commons often begin with one or more people who are motivated to start a project to fill a particular software need that they have. These projects usually start small in terms of development teams,[5] perhaps one or two people, who begin working on a software project.

During this pre-release period – what we call the Initiation Stage – the most important factors that lead a FOSS project to success – defined in this stage as a first public release of code – center around the attributes of the designated or *defacto* leader/developer of the project (who often may be the only developer on the project). Our analysis suggests that leaders who devote significant time and effort in the project are more likely to produce a first release. Other aspects of leadership that are influential in this stage include having a plan for current and future software architecture and functionality, having established goals, and continually providing good project documentation and maintaining a high quality web presence. These factors are important in the Initiation Stage because some projects get contributions from volunteers before the first release (the projects are visible on SF even when they have yet to produce a first release), and these attributes help to lay the foundation for later success in the Growth Stage.

Once a project achieves success and produces a first release, the story gets more complicated. We found in our data that about 15–20% of the successful growth projects are examples of Arrow 3 depicted in Figure 2. These projects are of interest only to small audiences and have small numbers of downloads as a result. However, our data reveal that most successful Growth Stage projects have over 1000 downloads and are of interest to 200 or more users.[6] The story

[5] Although there are examples of closed-source projects that get re-licensed and converted to open source projects – something West and O'Mahony (2005) refer to as "sponsored spinouts". The web browser Netscape, that eventually evolved into Mozilla and Firefox, is a high-profile example of this scenario.

[6] This is an estimate based on the 1000 downloads metric.

that our analysis uncovers is that once a project moves into the Growth Stage, the leadership skills of the project's development team, and the utility of the software itself begin to attract users. We have strong statistical support that often, at least one additional development team member joins the project and helps cause its continued success.

These FOSS commons tend to be governed by a benevolent dictator who manages the project as a form of common property. One or possibly a few leaders in the project tend to have authority over the version control system and determine what goes into the next release. Under this system of governance, the existing developers, users and potentially new development team members continue to make contributions and improvements to the software and related materials (e.g. documentation) and, in the successful collaborative instances, a virtuous circle is created. New versions are released, projects can continue to gain users, some of whom contribute back to the project (recall the importance of the community variables in the Growth Stage that we described earlier) and successful work continues. However, in other cases, particularly ones where some project developers have prior FOSS experience, our data suggest that these developers see the writing on the wall when the project is not achieving what it set out to achieve and eventually abandon the project.

Over the last year we have given several talks with software developers in the audience where we have described the "virtuous circle" story that has emerged from our data. After these talks we have had developers come up and confirm that the story we just told summarizes their own experiences. To these people, our findings may seem obvious. But we should remind readers that the story just described is grounded on careful, systematic quantitative data analysis and sizable datasets. The fact that the findings align well with what these practitioners have seen in their experience, provides us with an additional level of confidence in our results.

8. Conclusion: toward a general theory of Internet-based collective-action in digital information commons

In this paper we have tried to summarize a 5-year research project looking at factors leading to collaborative success or abandonment in one form of online digital information commons called free/libre and open source software or FOSS, a difficult task given all that we have to say. As we noted earlier, readers should care about FOSS as an online peer-production commons or, more accurately, an online peer-production common property regime, in part because computer programmers have been actively working in online collaboration for much longer than most of us. The collaborative principles found across successful FOSS projects can tell us much about how similar practices might be applied in other digital commons situations.

In the previous section, we described the basic story about how these projects work and evolve. Now, in conclusion, we will try and step back and close with

five important theoretical insights our study revealed that undoubtedly has implications for other FOSS-like digital information commons collaborations. These points should be of interest to anyone interested in digital information commons and online collective-action.

8.1. Motivations to contribute – A "Theory of Compound Incentives"

As we have stated, across both successful and abandoned projects, von Hippel's "user-centered need" is an important motivator for participation. In the early days of FOSS the vast majority of discussions focused on the volunteer nature of FOSS-collaboration. More recently, firms have joined in and encouraged employees to contribute. So von Hippel's user-centered need now captures the incentives of both volunteer developers and, in many instances, the motivations of firms. In addition, our research revealed that some developers participate in FOSS because of the enjoyment of using their skills for "serious leisure" (Stebbins 2001) or for the learning they gain by reading and editing other peoples' code (recall our opening story of Max, the 11-year-old programmer). In contrast to earlier FOSS work on motivations, we found in our survey data that the idea of showing off or demonstrating programming skill to the community was much less a motivator, and this is probably because the vast majority of the projects are small teams.

But what was most striking in our analysis (see Schweik and English 2012 for more detail) is that no particular motivation seemed to be aligned with either success or abandonment. But what we did discover in our contingency table analysis was that the higher number of different motivations reported by an individual respondent, the higher the success rates in both Initiation and Growth. This suggests that projects may be more successful if their developers have multiple reasons to contribute to the project (e.g. they need the software *and* they are paid to contribute).

8.2. FOSS Project Governance

Our findings suggest that a large majority of FOSS projects are governed simply through agreed upon social norms. This is the case because many of these projects involve very small teams. Programmers we interviewed often did not even recognize that they had rules for collaboration in place, and this was largely because many of the operational-level rules were embedded in the online software version control system that they use to store their current release and code they are working on for their future release. However, we also uncovered statistical evidence suggesting that to some degree projects move toward more formalized governance structures and operational rule systems as development teams get larger. In some cases, formalization means that very informal rules governing collaboration get slightly more formalized. Programmers appear to still seek systems of governance that minimize "friction", as Eric Raymond (2001) suggests.

A limitation in our study on the point of FOSS governance is that we explicitly tried to build a dataset that was a realistic representation of the population of FOSS projects, and undoubtedly, the vast majority of FOSS projects involve very small teams. Small teams need less formalized coordination. We hypothesize that had we sampled from the long tail of projects where large teams exist, we would most likely see higher levels of formalized governance structures.[7]

8.3. OSS and Group Size

In Schweik and English (2012, 74–76) we discuss three different theoretical perspectives about group size and collective-action. First, we make a connection between two famous theories proposed around the same time but in different fields: economist Mancur Olson (1965, 35) argued that "the larger the group, the less likely it will further its common interests". In a totally separate discipline, software engineering scholar Frederick Brooks ([1975] 1995, 25) theorized that that "adding manpower to a late [software] project makes it later". Both of these scholars made similar propositions: larger project teams will have more difficulties than smaller ones.

However, 30 years or more later, reflecting on collaboration related to the FOSS operating system Linux, self-proclaimed FOSS advocate Eric Raymond proposed "Linus' Law" that states: "Given enough eyeballs, all bugs are shallow" (Raymond 2001). In other words, in FOSS, larger development teams and active user communities help, rather than hurt.

In our data, we found that successful Growth Stage projects grow, albeit by a small number of programmers, but we also find strong evidence that the user community matters and is important. Our analysis lends strong support to Linus' Law over the Olson/Brooks theories in the context of online digital FOSS commons.

8.4. Face-to-Face meetings and Social Capital

In this paper we only touched on issues of social capital and FOSS commons, but we did in fact study it. One of the most interesting "non-findings" from our survey was that face-to-face meetings between multi-developer teams were not necessary conditions for achieving success. Many of our survey respondents working in projects classified as successful had rarely if ever met face-to-face with one or more developers on their team. But let us be clear; we are not saying that face-to-face does not matter or is not important in building social capital. It undoubtedly is. What we are saying is that some FOSS projects where collaborators are geographically distant can get around this by using

[7] Incidentally, in Chapters 5 and 7 of Schweik and English (2012) we build upon Ostrom's (2005) rule categories and analyze seven FOSS projects that all are connected via an overarching nonprofit foundation called OSGeo. In that work, we are attempting to move toward further systematic and comparative study of FOSS governance structures.

technologies such as Skype for virtual collaboration. It is not the same as meeting and having a beer together, but our findings suggest that having that beer with other collaborators is not a necessary condition for creating high levels of trust (which we found to be the case in most projects) and achieving collaborative success.

8.5. SF and Google as "power-law" intellectual matchmaking hubs

This question of team members and distance lead us to what we think is one of the most interesting and important findings of our study. In our analysis of the 2006 data, we discovered that SG projects tend to gain a developer, but at the same time, the vast majority of projects remain small teams (2–4 developers). In our 2009 SG survey, recall we asked questions about whether the project had gained a developer, and if so, whether that developer was co-located, geographically, or whether he or she was in the same city, same state, same country, same continent, or on a different continent. We discovered that of the multi-developer projects, more than 50% of them had a developer on a different continent.

This provides strong statistical evidence suggesting what may be a very important phenomenon that is important not only to FOSS but for any other digital information commons effort trying to mobilize collective-action. In fact, we have evidence that might very well explain underpinnings of collective action in, say, Wikipedia.

What we think we have learned is this: The vast majority of FOSS commons are not about creating and mobilizing large teams of developers (users, perhaps more so, developers not as much). What is happening is an "intellectual matchmaking" phenomenon, that is driven by the fact that SF acts a key "power-law" hub (Karpf 2010, 12) for FOSS software, coupled perhaps with search engines like Google. These are locations on the Internet-based where people go, looking to solve a user-centered need and are looking for a FOSS solution. The ease of which they can now find one or two or three other people working on a solution lead them into the virtuous circle we described earlier. They find a project of interest, engage, perhaps first as an end user, but over time, interact via the Internet and build social capital with the FOSS team, perhaps demonstrate an interest, a passion, and skills to contribute, and eventually are brought in as an additional developer. The story of FOSS-based collective action in our data is that it is not necessarily about growing large development teams; rather it is about people over sometimes large geographic areas discovering each other, and connecting with two or three other people with very similar user-centered needs, interests, passions and abilities. This power-law intellectual match-making phenomenon we just described undoubtedly occurs in other digital commons situations outside of FOSS. This could provide the foundation of a general theory of Internet-based collective action in digital information commons.

Literature cited

Aksulu, A. and M. Wade. 2010. A Comprehensive Review and Synthesis of Open Source Research. *Journal of the Association for Information Systems* 11(11/12):577–656.

Benkler, Y. 2006. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. New Haven, CT: Yale University Press.

Bollier, D. 2008. *Viral Spiral: How the Commoners Built a Digital Republic of Their Own*. New York: New Press.

Boyle, J. 2003. The Second Enclosure Movement and the Construction of the Public Domain. *Law and Contemporary Problems* 66(1–2):33–74.

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

Crowston, K., H. Annabi and J. Howison. 2003. Defining Open Source Project Success. Paper Presented at the Twenty-Fourth International Conference on Information Systems, Seattle, WA, December 14–17.

Crowston, K., K. Wei, J. Howison and A. Wiggins. 2012. Free/Libre Open Source Software: What We Know and What We Do Not Know. *ACM Computing Surveys* 44(2):1–35.

Deek, F. P. and J. McHugh. 2007. *Open Source Technology and Policy*. New York: Cambridge University Press.

Dafermos, G. 2005. Management and Virtual Decentralized Networks: The Linux Project. *First Monday* Special Issue No. 2. October 3.

David, P. A. and J. S. Shapiro. 2008. Community-Based Production of Open-Source Software: What Do We Know about the Developers Who Participate? *Information Economics and Policy* 20(4):364–398.

Drahos, P. and J. Braithwaite. 2002. *Information Feudalism: Who Owns the Knowledge Economy?* New York: The New Press.

English, R. and C. M. Schweik. 2007. Identifying Success and Tragedy of FLOSS Commons: A Preliminary Classification of Sourceforge.net Projects. *Upgrade: The European Journal of the Informatics Professional* 8(6):54–59.

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

Howison, J., M. Conklin and K. Crowston. 2006. FLOSSmole: a Collaborative Repository for FLOSS Research Data and Analyses. *International Journal of Information Technology and Web Engineering* 1(3):17–26.

Fogel, K. 2006. *Producing Open Source Software: How to Run a Successful Free Software Project.* Sebastopol, CA: O'Reilly Media.

Karpf, D. 2010. What can Wikipedia Tell Us About Open Source Politics? In *Proceedings of JITP 2010: The Politics of Open Source,* eds. Stuart Shulman and Charles M. Schweik, 2–30. Amherst, MA: University of Massachusetts. http://scholarworks.umasss.edu/jitpc2010/1/.

Krishnamurthy, S. 2002. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. *First Monday* 7(6).

MIT Media Lab. 2012. About Scratch. http://info.scratch.mit.edu/About_Scratch. (accessed July 25, 2012).

Olson, M. 1965. The Logic of Collective Action.

O'Reilly, T. 2003. *The Open Source Paradigm Shift*. http://tim.oreilly.com/articles/paradigmshift_0504.html. (accessed July 25, 2012).

OSI. 2012. http://opensource.org/licenses/category. (accessed July 26, 2012).

Ostrom, E. 1990. *Governing the Commons: The Evolution of Institutions for Collective Action.* Cambridge: Cambridge University Press.

Ostrom, E. 2005. *Understanding Institutional Diversity*. Princeton, NJ: Princeton University Press.

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

Riehle, D. 2007. The Economic Motivation of Open Source Software: Stakeholder Perspectives. *IEEE Computer* 40(40):25–32.

Robles-Martinezm G., J. M. Gonzalez-Barahona, J. Centeno-Gonzalez, V. Matellan-Olivera and L. Rodero-Merino. 2003. Studying the Evolution of Libre Software Projects Using Publicly Available Data. Paper Presented at the International Conference on Software Engineering, May 3–11.

Schweik, C. M. 2005. An Institutional Analysis Approach to Studying Libre Software Commons. *Upgrade: The European Journal for the Informatics Professional* 6(3):17–27.

Schweik, C. M. and R. English. 2012. *Internet Success: A Study of Open Source Software Commons*. Cambridge, MA: MIT Press.

Stallman, R. 1999. The GNU Operating System and the Free Software Movement. In *Open Sources: Voices from the Open Source Revolution*, eds. Chris DeBona, Sam Ockman and Mark Stone, 53–70. Sebastopol, CA: O'Reilly Media.

Stebbins, R. A. 2001. Serious Leisure. *Society* 38(4):53–57.

van Antwerp, M. and G. Madey. 2008. Advances in the Sourceforge Research Data Archive. Paper Presented at the Fourth int. Conference on Open Source Systems, Milan, Italy, 7 September 2008.

van Laerhoven, F. and E. Ostrom. 2007. Traditions and Trends in the Study of the Commons. *International Journal of the Commons* 1(1):3–28.

Von Hippel, E. 2005. *Democratizing Innovation*. Cambridge, MA: MIT Press.

Weiss, D. 2005. Measuring Success of Open Source Projects Using Web Search Engines. Paper Presented at the First International Conference on Open Source Systems, Genova, Italy, July 11–15.

West, J. and S. O'Mahony. 2005. Contrasting Community Building in Sponsored and Community Founded Open Source Projects. Paper Presented at the Thirty-eighth Hawaii International Conference on System Sciences, Big Island, January 3–6.

Wiggins, A. and K. Crowston. 2010. Reclassifying Success and Tragedy in FLOSS Projects. In *Proceedings of the Sixth International Conference on Open Source Software*, eds. P. Ågerfalk, C. Boldyreff, J. González-Barahona, G. Madey and J. Noll, 294–313. Berlin: Springer, Berlin.