2-2-2007

Tragedy of the FOSS Commons? Investigating the Institutional Designs of Free/Libre and Open Source Software Projects

Charles M. Schweik
University of Massachusetts Amherst

Robert English
University of Massachusetts Amherst

Tragedy of the FOSS Commons? Investigating the Institutional Designs of Free/Libre and Open Source Software Projects

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

Robert English
Center for Public Policy and Administration,
University of Massachusetts,
Amherst, MA USA
bobengl@gmail.com

NCDG Working Paper No. 07-003

Submitted February 2, 2007

A final version of this paper appears in First Monday, an online, peer-reviewed journal:
http://www.firstmonday.org/issues/issue12_2/

This material is based upon work supported by the National Science Foundation under Grant No. 0131923. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Tragedy of the FOSS Commons?
Investigating the Institutional Designs of
Free/Libre and Open Source Software Projects

By

Charles M. Schweik[1,2]
Robert English[2]

Draft – Please do not cite without permission from the authors

[1] Department of Natural Resources Conservation, University of Massachusetts, Amherst
[2] Center for Public Policy and Administration, University of Massachusetts, Amherst

Address all correspondence to:

Charles Schweik
Natural Resources Conservation
217 Holdsworth Hall
University of Massachusetts, Amherst
Amherst, MA 01003
e-mail: cschweik@pubpol.umass.edu
phone: 413 545 1824

Abstract

Free/Libre and Open Source Software projects (FOSS) are a form of Internet-based commons. Since 1968, when Garrett Hardin published his famous article "Tragedy of the Commons" in the journal *Science*, there has been significant interest in understanding how to manage commons appropriately, particularly in environmental fields. An important distinction between natural resource commons and FOSS commons is that the "tragedy" to be avoided in natural resources is over-harvesting and the potential destruction of the resource. In FOSS commons the "tragedy" to be avoided is project abandonment and a "dead" project. Institutions – defined as informal norms, more formalized rules, and governance structures – are mechanisms that have been shown to help overcome tragedies in some environmental commons situations. The goal of this paper is to more formally describe the concept of FOSS institutions and to conduct a preliminary examination of FOSS projects in order to shed light into institutions, their composition and importance to the projects. We report findings from an initial set of interviews of FOSS developers and find that in commons settings that need to encourage contribution rather than control over-appropriation, the institutional designs appear to be extremely lean and as unobtrusive as possible. To the FOSS programmers we interviewed, institutional structure adds transaction costs and hinders collective action. This is markedly different from traditional environmental commons settings.

Introduction

Much of the early literature related to Free/Libre and Open Source Software (FOSS) describes projects as teams of self-organized volunteers collaborating over the Internet to develop a software product. In recent years, however, we have witnessed a serious commitment by private firms, non-profit agencies and government agencies to embrace FOSS as an information technology strategy or policy (Hann, 2002; Myung, 2005; Becker, 2005; Peruvian Association of Free Software, 2005; Goldman and Gabriel, 2005). In some cases, not-for-profit foundations have been or are being established to provide financial support to FOSS projects and to provide some assurance that the software will be maintained and supported over time (see, for example, http://www.mozilla.org/foundation/). One could make the argument that, as a collaborative development paradigm, FOSS is maturing.

As more organizations become vested in FOSS, a central question will be how to increase the likelihood that these collaborations result in a "successful" product. As Steven Weber remarked (2004: 267): "one of the next steps in research on open source should be to build analytic models that try and specify the conditions that favor or hinder these experiments." We have argued elsewhere (Schweik, 2005) that to understand what leads to success or failure of FOSS projects one needs to look at a project's (1) physical attributes (e.g., type of programming languages used, communication and library management infrastructure, etc.), (2) community attributes (e.g., degree of user involvement, leadership characteristics, social capital, etc.), and (3) institutional design. The phrase "institutional design," as we use it, describes the informal norms and more formalized rules and governance structures that organize FOSS social interaction (Singleton, 1998; Nie and Ingram, 1998; Dietz,et al., 2003; Ostrom, 2005).

While there has been considerable research on FOSS in recent years, with some of it analyzing the physical or community aspects of projects, very little research exists that explicitly studies the third category: the structure or evolution of FOSS institutional designs.[1] Focusing on FOSS institutional designs draws attention to some FOSS questions that are currently not well understood: What do FOSS institutions look like? How important are they to the success or failure of a project? Are FOSS institutions more informal or formal? Are there some commonalities across FOSS projects in institutional structure? How do FOSS institutions evolve from project start-up to project "death"?

The goal of this paper is to more formally describe the concept of FOSS institutions and to conduct a preliminary examination of FOSS projects in order to shed light into institutions, their composition and importance to the projects. This paper is part of a longer-term project study a large number of FOSS projects looking for key factors that lead to success or tragedy in FOSS projects, and to understand project evolution over time (Schweik, 2005).

A Starting Point: The Tragedy of the FOSS Commons?

The phrase "collective action" in the social sciences describes situations where the efforts of more than one person are required to achieve a desired outcome (Sandler, 2004). A large proportion of FOSS projects found on hosting sites like Sourceforge.net – arguably the largest repository of FOSS projects – involve only one developer (Krishnamurthy, 2002). As Table 1 shows, as of April 2005, more than 90 percent of the projects listed on Sourceforge.net still involve less than five developers. By definition, projects with one developer could not be classified as collective action (assuming no one else is contributing to documentation or testing. Teams of 2-4 developers could be considered collective action situations, but they do not pose very difficult coordination problems, so they are not ideal

subjects for collective action research. While the statistics in Table 1 show that collaborations with a team size of five or more people are relatively small in number when compared to the total number of FOSS projects that exist, there are still a substantial number of FOSS projects that fall into these collective action, or in some cases global collective action, categories.

Table 1: Numbers of FOSS projects in Sourceforge.net organized by development team size (April, 25, 2005)

Source: FLOSSmole (2005)

Greater than 25 developers	224
Between 11 and 25 developers	1573
Between 5 and 10 developers	5532
Between 1 and 4 developers	86,373
Total number of FOSS projects on Sourceforge as of April, 2005	93,702

Collective action situations are usually connected to the creation or maintenance of public goods. Two characteristics define a public good: non-rivalry and non-excludability (Ostrom and Ostrom, 1977). Non-rivalry means that the use of the good by one individual does not reduce the amount of the good available for others to use. Non-excludability means that it is difficult to prevent people from utilizing the good. FOSS licensed software in digital form served on the Internet possess these properties. However, while FOSS can be considered a global public good, there are designated owners of the software (established, for example, through the copyright statements in the FOSS license, in comments of the code or on a project website) who act as gatekeepers to determine what can or cannot go into the project's next release. Viewed as a system of production, FOSS projects have been referred to as a "commons" (Bollier, 2002; Goldman and Gabriel, 2005), but, to be more precise, they are a "common property regime" (Schweik, 2005). (In this paper, we use "commons" and "common

property regime" as synonyms, and for readability we will use "commons").

There is a wide and rich literature on environmental commons (e.g., forests, fisheries, water systems) (See National Research Council, 2002 or Dietz et al., 2003 for summaries), and it is well understood that people involved in these situations face obstacles to social cooperation (Ostrom, 1990; Singleton, 1998; Ostrom et al., 1999; Dietz et al., 2003). A key problem of the commons is that individuals who cannot be excluded from the benefits of a good often have little incentive to contribute voluntarily toward the production or maintenance of that good – what are commonly referred to as "free-riders" (Olson, 1965; Sandler, 2004). In forest, fishery or water settings, the free-rider problem can lead to over-appropriation of the resource – the "Tragedy of the Commons" problem (Hardin, 1968).

FOSS commons, however, being digital, are distinctly and fundamentally different from environmental commons in that the potential tragedy is not one that results from over-appropriation of the resource (the software). In fact, the idea of a FOSS software being "over-harvested" – lots of people utilizing the software – would be considered a good thing, in that it would be gaining market share. So, if over-appropriation is not a problem, is there a potential tragedy of the commons in a FOSS context? Undoubtedly yes. Free-riders in this context are programmers, testers or documenters who utilize a particular FOSS software but do not contribute back in these capacities. In a FOSS setting, the tragedy of the commons comes when there are insufficient human resources available to continue to further develop and maintain the software and, as a result, the software project becomes abandoned (Schweik, 2005). The project fails to achieve the functionality and use that was perhaps envisioned when it began.

Three Important Attributes of FOSS Commons

In order to study FOSS institutions and their influence on the success or failure of projects, there are three important attributes that need to be considered: (1) Evolutionary Stage of the Project, (2) Development Team Size, and (3) Measures of Success or Tragedy. Short descriptions of each follow.

Evolutionary Stage of the Project. In earlier papers (Schweik and Semenov, 2003; Schweik, 2005) we noted that FOSS projects go through two stages, Initiation and Growth. We define the Initiation Stage as the initial period of time a project goes through where people are collaborating on software code but there has yet to be a public release of this code. We define the Growth Stage as the period after the first public release. Obviously, any empirical study of FOSS projects will find projects in each of these two stages. We suspect that the variables that influence success or failure (tragedy) in FOSS projects differ in these two stages.

Developer Team Size. Projects will also differ in the number of developers they have associated with the project (as shown in Table 1). From an institutional design standpoint, we expect FOSS projects with larger numbers of people collaborating to have more complex institutional designs. In addition, it is likely that the different developer team sizes in Table 1 probably reflect different stages of growth of the project. That is, many of the very large projects, in terms of developers, probably went through earlier periods where they had much smaller development teams. For this reason, we think studying these mid-range group sizes may be informative in terms of how FOSS projects evolve and grow larger.

Measures of Success or Failure. Research already exists that has worked to identify measures of success or failure of FOSS projects (Stewart and Ammeter 2002, Crowston, et al., 2003; Capiluppi et al., 2003). Building on these efforts, we think that measures of success or failure of FOSS commons

probably depend upon what evolutionary stage they are in. This means there are four possible outcomes: Initiation Stage success or failure and Growth Stage success and failure.

We define S*uccess in the Initiation Stage* as "a project producing a first public release." This can be easily measured by seeing if the project has produced a first release of the code.

As we see it, *Failure in the Initiation Stage* occurs when a project has not been able to produce a first release within the first year of the project being public. Preliminary data we have analyzed from Sourceforge.net indicates that projects over a year old that have not had a release are generally abandoned. This is easily measured since most project sites or hosting sites, like Sourceforge.net, provide information on when the project was started and when the first release was launched. Abandonment can be measured by (1) no code "commits" or changes in lines of code in the concurrent versioning system (CVS) or other repository over the course of a year, or (2) little or no activity on developer e-mail lists and forums over the course of a year. An often repeated saying about FOSS culture is "release early and often." Therefore, a one year timetable for this measure seems sufficient. (This was also supported in some of our FOSS developer interviews, described later).

We define a project as S*uccessful in the Growth Stage* if the project has produced "several releases of a software product that performs a useful computing task for at least a few users (it has to be downloaded and used)." The measure of "subsequent releases" is fairly easy to get for projects because this is stored on most project hosting websites. Measuring "a useful computing task" is harder and a bit more subjective. Acquiring the number of downloads recorded on project websites is probably the easiest measure, since this too is recorded on most project hosting sites. Other more time consuming measures to generate, include: (1) a content analysis of user forums or e-mail archives on utility of the software; (2) an actual download, installation and use (for some software); or (3) interviews with users

who have downloaded the product. The final parameter of this definition, "a few users" is also measured by the number of downloads.

Finally, we define *Failure in the Growth Stage* to be when a project appears abandoned (very few downloads, little development activity, such as code commits, going on, etc.) without achieving several subsequent releases.

We recognize that there will be some projects that will not easily fit into these categories. For example, an *Indeterminate in the Initiation Stage project* might be one that has yet to produce a first public release but shows significant developer activity. *Indeterminate in the Growth Stage* might be a project that shows development activity but has not yet produced several releases. Nonetheless, having a measurable definition of success and tragedy for various stages that can be operationalized for most projects is an important component for this and our broader FOSS research.

Institutions and Avoiding the Tragedy of the Commons

Institutions – informal norms, more formalized rules, and governance structures – are mechanisms that have been shown to help overcome tragedies in some environmental commons situations (Ostrom 1990; Ostrom et al., 1999; Dietz et al., 2003). For example, in her seminal book *Governing the Commons,* Elinor Ostrom[3] (1990) provides cases where local resource users have created their own self-governing systems which overcome free-rider problems and the over-harvesting situation that are the result of this behavior. Ostrom has emphasized in her work over the years that this collective development of institutions to govern natural resource commons is by no means easy. It takes hard work, to the point where she often refers to the local users as "artisans" who "craft" institutions. While the institutional designs across cases certainly vary, many evolve from initial interactions that establish trust and social capital, to the establishment of acceptable norms of behavior, to the

establishment of more formalized rules coupled with monitoring and sanctioning mechanisms for rule breakers (Ostrom, et al., 1999).

So, how might this work in a FOSS commons setting? Let us first adapt an example of informal relationship or norm building by Nee and Ingram (1998:25) and place it in a FOSS context. Suppose Dana, a software developer, establishes a new FOSS project, places it on SourceForge.net and invites others to participate. Suppose John comes along, who is interested in the project, as a developer or user, or both. John might send a number of e-mails to Dana about particular problems he is having with the software, or other questions related to learning how to contribute to the project. Dana answers these questions using time that could have been spent on other work. John reciprocates by bestowing on Dana a higher level of social approval. Both parties are rewarded by this exchange of assistance for approval. In addition, John is learning, and Dana is perhaps hoping that John might eventually help advance the software or at least help to promote it. Their exchange builds on mutual understanding and expectations that may be initially unspoken. Even if John is dependent on Dana's help, he does not want to give the impression of being "stupid" to Dana or to others who may be working on the project. Dana may expect John to eventually reciprocate by at least following informal norms of conduct and by providing some new code or enhancements to old code. She may also expect John to view and consider her as his "superior," at least in that she is the designated owner of the code. Nee and Ingram (1998:25) note that "[s]uch an implicit contract, an informal norm, may sooner or later be expressed in some communication in statements of expected behavior. Violation of the norm leads to such forms of punishment as anger or refusal to continue the interaction." For example, a common norm that can be violated (particularly by "newbies") in computing related issues is the norm often used in text messaging or e-mail shorthand: RTM (Read the Manual) or, in more anger, RTFM. If John continued to

ignore this norm and ask questions, this could lead Dana to terminate the relationship.

The above probably describes many FOSS collaborations, particularly ones in early stages and ones that are all-volunteer in nature. These informal norms that begin to be established are important because they reduce uncertainty in human interactions and help solve coordination problems, especially when a project reaches a situation where specialization and divisions of labor emerge (Nee and Ingram, 1998). And in cases where FOSS projects grow in terms of numbers of participants, or in cases where firms or government agencies contribute resources to the effort, we expect that these sets of informal norms will develop or evolve. For example, as the size of the development team increases, coordination norms, conventions or even decrees might develop and evolve to help the team coordinate their activities and solve coordination problems that may reoccur (Ullmann-Margalit, 1977).

Other FOSS researchers have noted the potential importance of institutions in FOSS projects. Crowston (2005) referred to these as "shared understandings" as he discussed the next steps of the broad, global FOSS research agenda. Goldman and Gabriel (2005: 232) state: "...every open-source project has some sort of governance because decisions must be made all the time" and suggest that the institutional structures get more complicated when firms become involved. They also suggest that governance and institutional designs change as the project evolves over time (Ibid., p. 233). And Weber (2004: 189) noted the importance of an institutional perspective when he said: "The open source process is an ongoing experiment. It is testing an imperfect mix of leadership, informal coordination mechanisms, implicit and explicit norms, along with some formal governance structures that are evolving and doing so at a rate that has been sufficient to hold surprisingly complex systems together."

General "Levels" of Institutions that may exist in FOSS projects

(Note: when we use the term "rules" in this section, we are referring to both *unwritten social norms and more formalized, documented rules*.)

Studies of the institutional design of natural resource commons have focused on three institutional "levels" (Ostrom et al., 1994; Ostrom, 2005; Schweik, 2005). The first level, the "Operational-level", is a general name for rules that influence the everyday decisions and actions made by project participants. In a FOSS setting, these are the norms or rules that specify how the further development and support of the software may proceed. The type of FOSS license used and the collaborative platform used for version control (e.g., CVS, subversion) establish some operational-level rules.

The second institutional level, "Collective-Choice," (Ostrom et al., 1994) is the generic name for two types of rules. The first type defines who is *eligible to undertake certain operational-level activities*. For example, in most projects there is probably some kind of norm or rule that specifies who has authority to promote or "commit" some code to the "next release" library. In some projects, this authority might be highly centralized; in other projects, the authority might be quite distributed, allowing each developer to promote their code when they feel it is ready. The other type of collective-choice rules specifies *who can change operational-level rules and the procedure to follow to make such a change*. For instance, as more developers join a project, there may be a need to change the operational-level rule on how code is reviewed before being promoted. Collective-choice rules would determine how a new operational procedure would be agreed upon and changed.

Constitutional-level rules are the third tier of institutions. *Constitutional-level rules specify who is allowed to change collective-choice rules and the procedures for making such changes*. One example

in a FOSS setting might be when the recognized leader of a project decides to move on to a new opportunity. Constitutional-level rules might specify who takes over in this person's absence. Another example is the case where a FOSS project operating entirely by volunteer developers becomes "embraced" by a commercial firm. This entry by a firm may (or may not) change who is involved in collective-choice processes.

The Negative Effects of Institutions?

Given the discussion above, it is likely that institutional configurations in FOSS commons will evolve as teams get larger and, at least in some cases, help the project avoid collective action problems. But Raymond (1999), a famous proponent of open source, raises an interesting puzzle regarding the free-rider problem in FOSS and the role institutions play in them:

> "The real free-rider problems in open-source software are more a function of *friction costs* [our emphasis] in submitting patches than anything else. A potential contributor with little stake in the cultural reputation game may, in the absence of money compensation, think ``It's not worth submitting this fix because I'll have to clean up the patch, write a ChangeLog entry, and sign the FSF assignment papers...". It's for this reason that the number of contributors (and, at second order, the success of) projects is strongly and inversely correlated with the *number of hoops* [our emphasis] each project makes a user go through to contribute."

Inferred in Raymond's statement is an understanding of the kinds of people who participate in FOSS development. We can classify FOSS developers into four types:

(1) relatively inexperienced *volunteer* programmers;
(2) inexperienced *paid* (by a firm, non-profit or government agency) developers;
(3) experienced, highly skilled *paid* developers; and,
(4) experienced and highly skilled *volunteer* developers;

For the first category of developers, volunteer and relatively inexperienced programmers, research has shown that there are strong motivators for participating in a FOSS development project,

with two of the primary motivations being economic: skill building and signaling. Inexperienced volunteer programmers often participate in FOSS projects in order to build their own skills through the reading and writing of code and by being subjected to the peer-review process. Through this participation, they hope to show off their skills, make connections and build a reputation with, in part, the hope of gaining future economic opportunities (Feller and Fitzgerald, 2001; Schweik and Semenov, 2003; Lerner and Tirole, 2005).

For the second and third categories of programmers – paid programmers – a primary motivation is obvious: they are doing what they are asked by their employer who pays them.

But the category of programmers Raymond refers to in the above quote falls in the fourth category in our list above: highly skilled, volunteer programmers who have little need or desire to further skill-build or signal their abilities to others. There is some evidence that in FOSS this fourth group of participants may be particularly important. For example, two studies of larger FOSS projects, Linux (Dempsey, et. al., 2002) and the GNOME desktop environment (Koch and Schneider, 2002), found that a smaller set of "core" developers contributed a majority of the code to these projects.[4] Ye and colleagues (2003) describe a situation where development of GIMP (the Gnu Image Manipulation Program), ceased for some time because there was no developer community to pick up the work when the initial developers decided to move on.

Consequently, Raymond is saying that the key free-rider problem in FOSS settings is how to get and keep *highly skilled volunteer programmers* to contribute their time and resources to the project. And the reference to the "number of hoops" in Raymond's quote suggests that the existence of too many established rules and procedures related to the operation of a project might be a factor that drives developers away.

Raymond's claims are supported by the work of O'Mahony (2003), which reports that FOSS developers resist centrally controlled governance and formal methods for organizing. And Holck and Jorgensen (2005: 2) summarize this nicely in their recent paper on "control" versus "anarchy" in FOSS projects: "developers would prefer loosely controlled projects with a flat hierarchy, relying on individual autonomy, tacit norms, and self-organization rather than commands, control, and explicit rules."

An Initial Empirical Study of FOSS Institutions

The above discussion leads to an interesting puzzle. On the one hand, there exists a rich literature on environmental commons situations that suggests that institutions – norms, rules, and governance structures developed by the participants themselves – are key to solving problems of collective action. Moreover, as Holck and Jorgensen (2005:2) report, traditional software development settings have relied on "diligent project management" including planning, directing and controlling the process. But on the other hand, studies of FOSS collaboration suggest that too much control, governance or other systems of organization might cause participants to stop collaborating, leading to project abandonment -- the tragedy of the FOSS commons. That is, to use Raymond's phrasing, institutions could create "friction" that leads to a collapse of collective action in FOSS settings.

This leads us to three fundamental research questions and three initial hypotheses:

Research Questions:

RQ1. What kinds of institutional configurations exist in FOSS projects?
RQ2. How do they evolve?
RQ3. What kinds of configurations appear to help, or hinder FOSS projects?

And at this juncture, we have three general hypotheses related to FOSS institutions:

Hypothesis 1: FOSS projects in the initiation stage (Figure 1) and/or involving a small number

of developers, will have institutional designs that are lean and informal. There may also be some sort of loose or informal, but understood, decision-making or governance structure established.

Hypothesis 2: FOSS projects will move incrementally from informal norms to more formalized rules and governance structures as more developers join the project.

Hypothesis 3: Projects with larger numbers of developers on their team and that are farther in the development lifecycle (the growth stage of Figure 1) will exhibit more formalized institutional structure than smaller projects, but there will be an effort to minimize the number of formalized rules and procedures so as to not alienate project members.

Research Methods

Given that we think that institutions evolve over time and this evolution is related to the number of developers on the project, we wanted to investigate cases that represent the four different "Number of Developer" categories presented in Table 2.

Table 2. FOSS PROJECT SAMPLING STRATEGY

	Number of Developers			
Stage	<5	>=5 and <=10	>10 and <=25	>25
Initiation	Cases 1,2	Case 4	Difficult to find	Difficult to find
Growth	Case 3	Case 5	Cases 6, 7	Cases 8,9

For this initial study, we limited ourselves to selecting two cases from each "number of developers" category for a total of eight cases. We had an opportunity to interview a developer from a third "less than 5 developer" project, which added one more case in that category (Table 2).

Two of the projects we studied were identified through personal connections with FOSS developers. Both of these were small development projects, falling in the "less than 5 developers" category. To identify two cases in the three other "number of developers" categories in Table 2, we utilized the FOSSMole (2005) data on Sourceforge.net projects to generate a case sampling database.

We eliminated projects with duplicate project names or with the number of developers field empty or set to zero. We then queried the database and organized the 93,702 projects left into four categories (presented earlier in Table 1). We utilized this database to randomly select cases to study, stratified by the number of developers, which is an attribute in the Sourceforge project metadata. We sent e-mails out to lead developers, asking them to be interviewed and then followed up on positive responses. The developers we interviewed represented projects from various FOSS software categories including: operating system components, content management platforms, web server and applications, utility programs, a Usenet news application, an instant messaging application and a Geographic Information System application. As noted in Table 2, identifying cases where there were larger groups of developers (e.g., > 10) and which were also in the Initiation Stage (as defined by not yet having a public release of the code) was difficult.

Since collective action is easier in small groups, we hypothesize that most projects tend to start small and grow in the number of participants as the software code base becomes larger and more complex (Sandler, 2004). The fact that it is difficult to find projects with greater then ten developers in the initiation phase lends credibility to this hypothesis.

We developed an interview protocol as part of the larger FOSS study outlined in Schweik (2005). This protocol consists of questions that cover (1) the history of the project; (2) ways to conceptualize success and failure in the project; (3) attributes about the software being developed; (4) collaborative infrastructure used; (5) attributes about the developers contributing to the project; and (6) general information about the institutional design of the project – the focus of this paper. In these interviews we did not specifically ask about the various categories of rules (e.g., boundary rules, scope rules, etc.) described earlier, but limited our questioning to the day-to-day operations of the project

(Operational rules, above). We also asked questions about who has authority to change those operational level rules ("Collective-Choice" arrangements, above) and asked whether there were higher level governance rules on, for example, how leaders or others in positions of authority in the project are replaced or whether a formal constitution for the project existed (Constitutional-level rules, above). The structured interview consists of about sixty questions in total and takes forty-five minutes to an hour to complete. Interviews were conducted over the phone, and several of our cases involved people working outside of the United States. Each interview was recorded using a digital voice recorder then transcribed and analyzed using Transana, an open source software for qualitative analysis. Transana allowed us to attach keywords to passages in the transcripts and group responses by question or by keyword. Finally, and importantly, several of the developers we talked to have worked on more than one FOSS project, so many of their answers reflected more projects than just the nine we report on in Table 2.

Results

Although the surveys we conducted covered aspects of open source collaboration beyond institutions, for this paper we will limit ourselves to the institutional aspect of the projects. Areas of discussion included: (1) operational-level rules in place; (2) how rule systems are learned by new participants; (3) how they handle rule breaking; (4) how conflict is dealt with; and (5) collective-choice and constitutional level aspects.

Operational-level rules. In our interviews, we first asked interviewees to describe the important operational rules in place for their project. Several interviewees from the three smallest projects (less than 5 developers) reacted initially by saying "we have no operational rules" or something similar.

Further discussion revealed there were some procedures in place, but, other than the requirements of the FOSS license, not many had been formalized. For example, related to an operational rule about modularity of code, one interviewee of a small project said: “if you send [one of the lead developers] a large patch... he'll outright reject it. He'll tell you to cut it up in 6 or 8 pieces so he can understand it completely before going on to the next piece.” But what operational rules these small projects had in place were clearly few, and very informal in nature.

Similarly, the two projects we studied with between 5 and 10 developers also reported very few operational level rules, but the respondents did report having informal rules established to help coordinate the work effort. Like the smaller projects, these rules were not formally documented. As one respondent put it: “Some operational-level rules are documented in the project forum, but others are just common sense.”

As we move to the next project size category – development communities with between 10 and 25 developers, the degree to which operational rules have been considered changes, as does the informal or formal nature of those rules. For example, in one project, an interviewee who was the designated project lead said this:

> “The important operational rules are...that, ... with some help from my project manager, [we] maintain the list of people who are allowed to make changes to the code. They're expected not to make changes that will be controversial without first discussing them with the development team, usually via our development mailing list. Those people have access to managing our bug reporting system as do some other people. All of them are expected to act professional, in a respectful manner, towards users, even when a user seems to not be very reasonable, or aware of what's going on. Really, it's sort of mostly a trust based system, it's worked very well with a few minor exceptions, which have been taken care of by saying to the person: `look, you need to not do this again, or you're not gonna be allowed to continue to contribute."'

Similarly, an interviewee in the other project in this category reported a move from informal to more formal procedures because of coordination problems that had occurred in the past:

> "It's a consensus model, a non-written agreement how to work [together]..., and this is something [that is] partially written down for [the process of] releases. Those [exist] because we have had problems in the past where some steps were missing, so we had to redo the release, which is, of course, something unfortunate."

Finally, one of the two projects with more than 25 developers reported mostly informal operational level rules understood by most participants, coupled with some documented rules or project guidelines (such as standards for writing code) on the project Wiki. When we, for clarification, repeated our understanding of this to the interviewer, and he responded by saying:

> "That's right. Remarkable that it still works, isn't it?"

The second project with more than 25 developers reported the most extensive formal documentation of operational-level rules. For this project, substantial documentation exists on the project website, including documents that explain how people can participate in code development, testing, writing documentation, etc. Other documents describe the project's coding standards or how to implement "secure code", and there exists a developer's guide and handbook. Finally, written documentation describes how to avoid the situation of creating too many modules that are similar in nature that causes, in their words, "confusion, clutter and inefficiency." But with this formally written library of documents established, an interviewee on the project responded: "Basically the operational rules are pretty simple. It's read the documentation, [find the component] you want to work on, and if you've got questions you have the mailing list available."

How are rules learned? When we asked interviewees of all project sizes how new participants learn operational rules, very similar responses were given: They learn the rules by watching the discussion in project forums (usually e-mail lists), getting a sense from the observation on how to

proceed and perhaps having someone on the existing team explain things to them. Then they go ahead, and learn by doing. Three of the four larger projects noted that someone on the team helps guide new developers or that new developers are instructed to read documentation written to help new developers join in.

Rule breaking. We followed this discussion with a question about what happens if an operational rule is broken by a team member and whether any formal procedure is in place to handle rule-breakers. For example, in environmental commons it has been shown that a system of "graduated sanctions" helps deal with rule-breaking situations (Ostrom, 1990, Dietz, Ostrom and Stern, 2003). Eight of the nine projects we reviewed had no system in place to handle rule breaking. Small projects had really never run into a situation where they needed to handle this problem. One of the participants in a larger project put it this way:

> "The closest thing to breaking a rule would be checking in broken or poor code. It is handled very graciously and not in an accusatory way. It's not like a public berating or anything."

Another said:

> "[This is handled] either one-on-one [between the project lead or someone of authority and the rule breaker] or via the list. But a very cooperative situation. Small hand slap and that's it."

However, an interviewee of one of the largest projects was a little tougher:

> "People call them out. You know, you get flamed on the mailing list. That's generally how it happens."

Conflict management. Having established some kind of conflict resolution mechanism has also been seen in environmental commons situations as something important. In our interviews, we asked respondents whether they had witnessed conflicts and whether there were any rules or procedures for

resolving such conflicts. In the three smallest projects this was not even seen as an issue worth discussing. In all other cases (projects greater than 5 developers all the way up to the projects with greater than 25 developers), conflicts were usually resolved through discussions over the project forums, with most being resolved through consensus. In one large case the interviewer said: it is “rarely the case where two people are at the same level of competence for a problem and they have completely different opinions.”

In a few cases the lead developer needed to talk to one of the participants separately to help resolve the problem. And three of the four largest projects reported having past situations where a conflict resolved through consensus led to the person whose idea wasn't taken leaving the project. Two of these led to efforts to “fork” the project (take the code and start a new project using that code). However, in neither case was it reported that this fork was successful.

Collective-choice and Constitutional-level rules. Earlier, we introduced two other levels of rules: Collective-Choice and Constitutional. Recall that Collective-Choice rules define who is eligible to undertake certain operational-level activities (e.g., the governance hierarchy) and also specify who can change operational-level rules and the procedure to follow to make such a change. Constitutional-level rules specify who is allowed to change collective-choice rules and the procedure for making those changes. Interviewees were asked questions related to these concepts.

In the projects with developer teams of 10 or fewer people, the governance structure or hierarchy was very flat. Most projects (including our largest project of over 25 people) had one developer who was seen as the lead and had the authority to work with the next release library, and everyone else worked with or considered themselves “reporting into” that lead developer, even if such a hierarchy was not formally defined. The other four projects we interviewed with teams of 10 or more developers,

exhibited slightly more structure. One of the projects between 10 and 25 developers reported a flat hierarchical structure but had a designated lead developer and a "project manager" to help coordinate the effort. In the other case in this category, the lead developer was seen as a kind of generalist, with others on the team being more specialists of certain components of the project. These people were assigned to different components by their capabilities and expertise, and certain people could prove themselves very competent and gain informal authority. This describes the kind of "meritocracy" that is often referenced in literature about open source projects.

The other project with more than 25 people had a designated project lead, with several sub-leaders of various major components of the project. After these leadership positions, however, most other developers were generally considered the same level, although some had more skills or knowledge than others. As our interviewee put it:

> "There is [sic] definitely people whose job it is to project manage, to make sure projects are getting done... but... basically every developer out there is to some extent a project manager... we have people in well-defined roles and more responsibility than others for sure, but in general its not like the hierarchical tree of management. It's not like we have worker bees and then bosses."

To our surprise, the other project with over 25 developers reported a very informal hierarchy. People on the project know who is the authority on the project for various components, but, as this respondent put it, there is no "explicit identification of individual roles that have been outlined or publicly demonstrated on the web..." This case also raised an interesting issue regarding project management and the role of developers paid to work on a project by a company with some interest in the project. In this case, the project's name actually referenced the company in it, so the tie between the software and the company was clear. Further, several of the project participants were employees of the interested company, including our interviewee, who referred to himself as a "key architect" on the

project. What this respondent found interesting, however, was that often other project participants were confused about the actual authority or influence the company had on the project, regardless of the company's explicit relationship. The respondent said frankly that the project wasn't hugely important to the company, but wanted it to keep going. In short, our interviewee summarized it this way:

> "When a company sponsors an open source project they need to realize that people are going to perceive [that the company is]... far more involved in [the project] than it might actually be.

Regarding the question of how operational-level rules are changed, we found that in most of the projects, large or small, the person seen as the project lead had most of the authority. It was only in the larger projects where respondents reported that a team discussion was needed before any major operating rule was changed. For instance, in one of the largest projects, the interviewee noted that

> "operational rules [change or are added] when something is not getting done that is needed to be done, and someone takes up the responsibility to do it."

We concluded our questions on the institutional design of these FOSS projects by asking interviewees about the constitutional level: Who can change the process in which operational rules are crafted? Who has the authority to do this? A good example of what we mean by this might be the existence of a "board of directors" in a non-profit organization since board of directors have some authority over the lead of a project and how they manage or change operations. We also asked to what extent contractual arrangements were in place on the project. Given what we heard about the Collective-Choice level rules, we were not surprised to hear that most projects had very few, if any, formal constitutional level rules in place. Even in the large projects, for example, there were no formal procedures on how one would be chosen to replace someone of authority who was leaving the project.

In this example, most cases reported that there would be probably a logical choice based on merit and expertise, perhaps coupled with some public discussion in the forums.

Two of the larger projects, one in the greater than 10 category and the other in the greater than 25 category, reported a more complex situation with regard to their working relationship with nonprofit organizations. The largest project worked closely with both a nonprofit and a private company, both using and promoting the core software. The nonprofit is just putting in place a governing board, and the private company is owned by the employees. So, while constitutional-level rule systems were not really in place, the respondent thought in the future some might eventually be there based on these relationships.

One other interviewee working with a project in the greater than 10 developer but less than 25 category, reported that in the past year his project had "joined" in a kind of federation of FOSS projects working toward a broader and mutually reinforcing goal. The result was the establishment of a foundation that acts as an umbrella organization for eight or more related, but separate FOSS projects. This foundation has an oversight board and is imposing a few rules on related projects, should they want to be considered part of this "federation" (our term, not theirs). One requirement for membership is that each associated project must have a steering committee in place. According to our interviewee, there is very little interest by project participants to meet this requirement. They prefer the informal mode of operations, with no formal committee or governing body. This is an issue they are still dealing with, and it is not clear what the result will be.

Discussion

Earlier we presented some research questions and some initial hypotheses related to the institutional designs of FOSS commons. The analysis above of nine FOSS projects with differing

"developer size classes" leads to several preliminary conclusions that tend to support our hypotheses:

FOSS institutions appear to evolve, as expected. Projects with smaller sized development teams and earlier in their lifecycle tend to have very lean institutional components and operate through informal systems and rules.

As projects gain developers, they continue to exhibit fairly lean sets of operational level rules. And many of the operational-level rules that exist are in part established by the collaborative platform the team uses to coordinate their work (such as the Concurrent Versioning System (CVS) that is used by many FOSS development projects). This differs significantly from environmental commons, for in the environmental domain, there are not usually Internet-based technologies that are used to coordinate collective action. This finding raises an interesting question: To what degree does the design of the versioning system and other technologies influence or change the way a FOSS team collaborates? Does system design affect the performance of the team? Literature on virtual teams suggests that collaborative infrastructure is an important factor (Gibson and Cohen, 2003; Kelly and Jones, 2001; Dube and Pare, 2001). And our finding that many operational-level rules actually are embedded within the collaborative platform suggests this to be true in FOSS as well. However, in our interviews we were generally told that the versioning system really didn't contribute much to the success or failure of the project. We also learned that a wide variety of technologies (e.g., email listservs, Internet Relay Chats, bug tracker applications) were used across projects to coordinate work. So it appears to be still an open question as to the influence of collaborative tools on project success or failure.

There was more attention to Constitutional and Collective choice processes in the larger developer communities (>10 developers) compared to the smaller communities (<10 developers). But again, in all projects these sets of rules were also mostly informal mechanisms and were relatively thin.

And this leads us to perhaps most important conclusion: Institutional designs across all development team sizes appear to be kept as lean as possible. Our analysis here supports the statement by Raymond about keeping "friction costs" to coordination low. This is a relatively stark distinction between FOSS commons and the more traditionally studied environmental commons. We think this difference is best explained by the fundamental difference in the type of "tragedy" that could occur: over-harvesting in the environmental case, compared to under-production in the FOSS development case. In a commons that needs to encourage contributions rather than control over-appropriation, institutional designs need to be in place to help coordinate collective action, but need to be as unobtrusive as possible. We kept hearing the desire of participants to code and not get caught up in coordination costs that formal rule systems bring with them. Part of this might be attributed to a "culture" in FOSS, but our impression from our interviews is that the formal structure is simply not needed. To these programmers, structure adds transaction costs and hinders collective action.

Conclusion

In this paper we have emphasized a number of points. First, while the vast majority of FOSS projects involve very small teams or even one individual, as of April 2005, in the SourceForge repository alone, there were over 1700 projects with eleven or more developers (Table 1). And given this is data from only one (although arguably the largest) FOSS hosting site, this likely underestimates the number of FOSS projects with relatively large development teams.

Second, we no longer think of FOSS development projects as being comprised entirely of volunteer collaborators. Firms, nonprofit organizations and/or government agencies pay some FOSS developers. Three of the projects in our small sample had some level of participation by at least one of

these types of organizations.

Third, we emphasized that FOSS software projects are an Internet-based "commons" or more precisely, a "common property regime" working to produce a public good. FOSS software have designated owners (via copyright) and these projects have gatekeepers who can rightfully keep people out of participating in development.

Fourth, we noted that there is a wide and rich literature studying commons situations, much of it studying commons in natural resources, and much of the attention has been on problems related to collective action.

Fifth, we noted that a key difference between environmental commons and FOSS commons is the kind of "tragedy of the commons" they face. In environmental settings it is over-harvesting of the resource. In FOSS commons, the tragedy is an under-production or maintenance problem.

Sixth, we noted that it is well-known in environmental commons that institutional designs – norms, formal rules and governance structures – often help to overcome commons tragedies. We noted that some FOSS research suggested that too much governance structure and rules to the collaboration may get in the way of FOSS collaborations.

Seventh, we argued that there isn't a great deal known about FOSS project institutions or how they evolve over time.

The rest of the paper set out to investigate FOSS institutional structure and get a sense of their evolution by analyzing a small number of FOSS projects. We studied nine projects, stratified by developer team size.

Perhaps the most important finding of this study is that the rules and governance structures established in these projects were mostly informal (meaning articulated in the form of social norms, and

not formally documented) and the organizational structures are quite flat. This was especially true in the smaller (group size) projects. The four larger projects (greater than 10 developers) exhibited a bit more formality in operational rules and governance structures, but in all cases they tried to keep these systems as simple and lean as possible. In many cases, the operational rules that existed were largely driven by the collaborative platform they used to coordinate their activities. This lack of formal institutions, particularly in the larger projects, was a bit of a surprise to us given the importance of institutions in natural resource commons settings and is an interesting finding to students studying commons and collective action issues. The very different "tragedies" in natural resource commons compared to FOSS commons – over-appropriation in the former, under-production in the latter – is probably a key reason for this finding. In environmental commons, institutions (including monitoring and sanctioning mechanisms) are required to force people to comply (e.g., not free-ride) in an effort to avoid over-use of natural resources. In FOSS settings, institutions and formal governance appear to be viewed as a barrier to free creativity and innovation.

Another interesting finding was that the projects of different size classes did exhibit the institutional change patterns we expected – moving from very simple norms of behavior to more complex (but often still relatively informal) structures – as we moved from studying small groups to large ones. This suggests that one way to understand the trajectory of FOSS projects might be to study different size classes rather than having to search through (perhaps non-existent) historical records of one particular project or to locate and interview participants who may have long gone.

But clearly, results from this small set of FOSS cases is not generalizable. Working in collaboration with the FLOSSMole project, we are in the process of developing a larger database of FOSS cases (from Sourceforge.net as well as other hosting sites) where we will be able to do an

extensive on-line survey and eventually quantitative analyses that get at, in part, FOSS institutional structure and change. The work we present here on institutional aspects (and other information gleaned from these initial interviews on physical and community aspects of these projects) is helping us develop a general theory of success and failure of FOSS projects which can then be tested in the next stage of our study.

Acknowledgments

Support for this study was provided by a grant from the U.S. National Science Foundation (NSFIIS 0447623). However, the findings, recommendations, and opinions expressed are those of the authors and do not necessarily reflect the views of the funding agency. The authors would like to thank Megan Conklin, Kevin Crowston and others at the FLOSSmole project (http://ossmole.sourceforge.net/) for making their Sourceforge data available for analysis. Finally, we appreciate the comments by Michelle Sagan of the National Center for Digital Government (www.ncdg.org) and Mike Hammel on previous versions of the manuscript.

References

D. Becker. 2005. CNet News.com: "California considers open source shift," at http://news.com.com/California+considers+open-source+shift/2100-7344_3-5327581.html, accessed 20 September 2005.

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

A. Capiluppi, P. Lago, and M. Morisio, 2003. "Evidences in the Evolution of OS projects through Changelog Analyses," In: J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (editors). *Taking Stock of the Bazaar: Proceedings of the 3rd Workshop on Open Source Software Engineering,* at http://opensource.ucc.ie/icse2003.

K. Crowston, H. Annabi, and J. Howiston, 2003. "Defining Open Source Project Success." In: *Proceedings of the 24th International Conference on Information Systems (ICIS 2003)*. Seattle, WA.

K. Crowston, 2005. "Future Research on FLOSS Development." *First Monday,* volume 10, issue 10, at http://www.firstmonday.org/issues/special10_10/crowston/

B.J. Dempsey, D. Weiss, P. Jones, and J. Greenberg, 2002. "Who is an Open Source Developer?" *Communications of the ACM*. volume 45, issue 2, pp. 67-72.

T. Dietz, E. Ostrom, and P. Stern, 2003. "The Struggle to Govern the Commons," *Science*, 302: 1907-1912.

M. Divitini, L. Jaccheri, E. Monteiro, and H. Traetteberg, 2003. "Open Source Processes: No Place for Politics?," In: J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (editors). *Taking Stock of the Bazaar: Proceedings of the 3rd Workshop on Open Source Software Engineering*, at http://opensource.ucc.ie/icse2003.

L. Dube and G. Pare, 2001. Global Virtual Teams. *Communications of the ACM*, 44(12), 71-73.

J. Feller and B. Fitzgerald, 2001. *Understanding Open Source Software Development.* London: Addison-Wesley.

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

C.B. Gibson, C. B., and S.G. Cohen, 2003. Virtual teams that work :Creating conditions for virtual team effectiveness (1st ed.). San Francisco: Jossey-Bass

R. Goldman and R.P. Gabriel, 2005. *Innovation Happens Elsewhere: Open Source as Business Strategy*. Amsterdam: Elsevier

R. W. Hahn, 2002. *Government Policy toward Open Source Software.* AEI-Brookings Joint Center for Regulatory Studies. Washington, DC.: Brookings Institution Press

G. Hardin, 1968. "The Tragedy of the Commons." *Science*. 162, pp. 1243-48.

S. Kelly and M. Jones, 2001. Groupware and the Social Infrastructure of Communication. *Communications of the ACM*, 44(12), 77-79.

S. Koch, and G. Schneider, 2002. "Effort, Co-operation and Co-ordination in an open source software project: GNOME." *Information Systems Journal*. volume 12, number 1, pp. 27-42.

S. Krishnamurthy, 2002. "Cave or Community?: An Empirical Examination of 100 Mature Open Source Projects", *First Monday*, volume 7, issue 6. at http://firstmonday.org/issues/issue7_6/krishnamurthy/

J. Lerner and J. Tirole, 2005. "Economic Perspectives on Open Source." In: B. Feller, B. Fitzgerald, S.A. Hissam, and K.R. Lakhani (editors). *Perspectives on Free and Open Source Software.* Cambridge, MA.: MIT Press.

S. Myung, 2005. ZDNet UK, "South Korean govt goes open source." at http://news.zdnet.co.uk/software/linuxunix/0,39020390,39116799,00.htm, accessed 21 November 2005.

National Research Council, 2002. *The Drama of the Commons,* Committee on the Human Dimensions of Global Change. Washington, DC.: National Academy Press.

V. Nee and P. Ingram, 1998. "Embeddedness and Beyond: Institutions, Exchange, and Social Structure," In: M.C. Brinton and V. Nee. (editors). *The New Institutionalism in Sociology.* New York: Russell Sage Foundation

M. Olson, 1965. *The Logic of Collective Action*. Cambridge: Harvard University Press.

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

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

E. Ostrom, J. Burger, C.B. Field,.R.B. Norgaard, and D. Policansky, 1999. "Revisiting the Commons: Local Lessons, Global Challenges." Science 284. pp. 278-282.

E. Ostrom, R. Gardner, and J. Walker, 1994. *Rules, Games and Common-Pool Resources*. Ann Arbor: University of Michigan Press.

V. Ostrom and E. Ostrom, 1977. "Public Goods and Public Choices," In: E.S. Savas (editor). *Alternatives for delivering Public Services: Toward Improved Performance*. Bolder CO.: Westview Press. pp. 7-49.

Peruvian Association of Free Software, "Peruvian congress approves law in favor of free software," at http://www.apesol.org/news/197, accessed on 28 September 2005.

E. Raymond, 1999. "The Cathedral and the Bazzar: Musings on Linux and Open Source by an Accidental Revolutionary," at http://www.ora.de/catalog/cb/chapter/, accessed 5 February 2006.

T. Sandler, 2004. *Global Collective Action*. Cambridge: Cambridge University Press.

C. M. Schweik, 2005. "An Institutional Analysis Approach to Studying Libre Software "Commons". Upgrade" *The European Journal for the Informatics Professional.* (June), at http://www.upgrade-cepis.org/issues/2005/3/up6-3Schweik.pdf.

C. M. Schweik and A. Semenov, 2003. "The Institutional Design of "Open Source" Programming: Implications for Addressing Complex Public Policy and Management Problems." *First Monday,* volume 8, number 2, at http://www.firstmonday.org/issues/issue8_1/schweik/.

M. Shaikh and T. Cornford, 2003. "Version Management Tools: CVS to BK in the Linux Kernel," In J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (editors). *Taking Stock of the Bazaar: Proceedings of the 3rd Workshop on Open Source Software Engineering*, at http://opensource.ucc.ie/icse2003.

S. Singleton 1998. *Constructing Cooperation: The Evolution of Institutions of Comanagement*. Ann

Arbor, MI.: University of Michigan Press.

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

E. Ullmann-Margalit, 1977. *The Emergence of Norms*. Oxford, England: Clarendon Press:

R. Van Wendel de Joode, J.A. De Bruijn, M.J.G. Van Eeten. 2003. *Protecting the Virtual Commons; Self-organizing Open Source Communities and Innovative Intellectual Property Regimes*. The Hague: T.M.C. Asser Press.

S. Weber. 2004. *The Success of Open Source*. Cambridge, MA.: Harvard University Press.

Endnotes

1. This is not to say that there is no work looking at FOSS institutional components. See for example, Van Wendel de Joode et al. (2003). And Holck and Jorgensen's (2005) study Mozilla and FreeBSD provides an excellent description of institutional components of these projects.

2. We should note that another trajectory or set of trajectories we do not list here are the cases where commercial, "closed source" software is relicensed by a firm or organization that developed it as FOSS. In these instances, the FOSS project might start out with a significant amount of code and maybe a first release. The number of developers associated with the project could be very few, or many, depending on if the firm decides to continue to pay staff to support the code. For this paper, we will not focus on this type of case, but we recognize that it exists.

3. For readers who may be unfamiliar with Elinor Ostrom's work – she is one of the premier social science scholars who has devoted a lifetime to studying how humans organize in collective action and commons settings – mostly (but not entirely) in environmental management settings.

4. Sandler (2005) refers to these situations as "weighted sum aggregation", where some people contribute more than others toward the production of public goods.

5. This project is part of a larger project studying factors that lead to success and failure of FOSS projects. The study of FOSS institutions is one component – albeit an important one – of this study. See Schweik (2005) for more information.