On the Intrinsic and Extrinsic Motivation of Free/Libre/Open Source (FLOSS) Developers

Sandeep Krishnamurthy

Motivation in the context of open source software may be seen as fundamentally different due to the presence of unpaid programmers, implicit rather than explicit forms of control and a different methodology for software development. Since software development is a creative task, the motivation of open source programmers can be compared to individuals in creative industries (Caves 2002). This paper summarizes the important trends in the research on motivation in open source and identifies variables that should be included in future research. Specifically, the current literature favors a taxonomy that considers two components of motivation—intrinsic (e.g., fun, flow, learning, community) and extrinsic (e.g., financial rewards, improving future job prospects, signaling quality). I make a case for incorporating both elements in developing an integrative theory about developer motivation. Three elements are identified as being unique to FLOSS development- diversity of project structures, co-existence of companies and communities and co-existence of creative and commercial elements. The important empirical evidence on FLOSS developer motivation is presented and analyzed. Four factors are identified as important mitigating and moderating factors in the conversation surrounding developer motivation- financial incentives, nature of task, group size and group structure. The role of these factors on developer motivation is discussed.

"For me work is the oddity. Work is a kind of compromise. I do work which is as close as possible to my passions to make working tolerable. But I feel most myself when I am doing this open source stuff. When I am doing this and give it my complete and full attention then everything else around me fades away and dissolves and I become completely focused."

Seb Potter, Pro-Am open source programmer[1]

Sandeep Krishnamurthy is Associate Professor of E-Commerce and Marketing at the University of Washington. He is passionately interested in all aspects of open source software development and has written extensively on the topic. He is also the author of two books on E-Commerce. He welcomes you to visit his web page (<http://faculty.washington.edu/sandeep>) and his blog (<http://sandeepworld.blogspot.com>).

Knowledge, Technology, & Policy, Winter 2006, Vol. 18, No. 4, pp. 17-39.

Introduction

The motivation of FLOSS developers has been a matter of great discussion, debate and scholarly inquiry in the academic literature on the topic. In an early letter to hobbyists Bill Gates has said:[2]

> "Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all bugs, documenting his product, and distribute for free?" (Moody 2001).

Following Lakhani and Wolf (2005), Lakhani and Von Hippel (2003) and Lerner and Tirole (2004), the current scholarly thinking favors a taxonomy that considers two components of motivation.[3] One component is intrinsic (i.e., deriving from the very act of participation, e.g., fun, flow, learning, community), the other is extrinsic (i.e., deriving from external rewards, e.g., financial rewards, improving future job prospects, signaling quality). It is the contention of this paper that these two components are frequently viewed as being independent of each other and an integrative view is needed to gain a deeper understanding.

Consider this scenario. A corporation has provided a considerable reward (or bounty) for open source developers that is contingent on the completion of a certain piece of software development.[4] Developers compete for these rewards and only one entry wins. What is the motivation of the actors in this scenario? Are individuals who submit code to be considered extrinsically motivated because they were motivated by the potential reward offered by the corporation? Could they also be drawn to the complexity of the software development task, thus, making them somewhat intrinsically motivated? Does the size of the reward truly affect the quality of work? For only certain subsets of developers? For projects with certain characteristics?

These questions point out the subtleties of FLOSS developer motivation. While many FLOSS developers are intensely motivated to solve problems or "scratch their own itch," the place for direct and indirect, immediate and delayed financial rewards is real. One study found that Apache developers "earned total wages of $82,000, with a mean of $80,000 for people with no rank or with rank developers and a mean of $110,000 for people with rank of committer" (Hann, Slaughter, Roberts and Fielding 2002). IBM has invested at least $1 billion in Linux-related operations. Red Hat's revenue in the year 2004 was $124.7 million. Moreover, developers who work on FLOSS projects come from different backgrounds and are motivated by different things. While some value financial rewards highly, others may be more interested in learning a new area or solving a tricky problem. For instance, Hars and Ou (2001) report that altruism was more closely tied to effort for student programmers, but not for paid programmers.

It is the thesis of this paper that that it would be injurious to this academic field if scholars took sides on the intrinsic-extrinsic divide.[5] Rather, what all involved need to recognize is the multi-faceted nature of developer motiva-

tion. Disparate findings (e.g., Lakhani and Wolf (2005) find evidence in favor of intrinsic motivation while Roberts, Hann and Slaughter (2004) find that intrinsic motivation does not affect participation levels) need to be reconciled and a richer theory of developer motivation needs to emerge. Rather than viewing intrinsic and extrinsic components as "either-or," scholars must consider both as important and having different forms of value. Indeed, the interaction between intrinsic and extrinsic motivations provides a rich area of scholarly inquiry. For instance, there is some evidence now that providing extrinsic rewards may *not* crowd out internal motivations, as was initially thought (Roberts, Hann and Slaughter 2004).

What Makes Open Source Unique?

Diversity of Project Structures

The open source space is exceedingly diverse along multiple dimensions. As a result, frequently, the literature focuses on one subset of the space ignoring the others. What is right for one subset may not work for another. Then, it is appropriate to ponder if it is possible to identify one theory of motivation among the population of all developers who work on any open source project.

Here are a few source of diversity among projects:

- One key source of the diversity of open source projects is group size. There are many projects that are large (e.g., Linux, Apache). However, a large variety of them are small by design or "caves" (Krishnamurthy 2002). Members of smaller groups may be motivated differently.

- There are three types of FLOSS groups—one in which all developers are unpaid, a second in which some developers are paid, and a third in which all are paid. Motivational theories for one type of project may not transfer easily to another type.

- Groups are in different stages of development. Consider the table below, which categorizes all the projects hosted on Sourceforge at two different times.[6]

Before we discuss this data, a caveat. Sourceforge lets project leaders assign a label to their project and does not define them. Hence, it may well be

		Data on Dec 30, 2004		Data on Jan 13, 2005	
	Development Stage	**Number**	**Percent**	**Number**	**Percent**
1	Planning	14,716	22.42 %	15,049	22.26 %
2	Pre-Alpha	11,185	17.04 %	11,493	17.00 %
3	Alpha	11,310	17.23 %	11,644	17.23 %
4	Beta	14,380	21.90 %	14,799	21.89 %
5	Production/Stable	11,975	18.24 %	12,399	18.34 %
6	Mature	1,077	1.64 %	1,110	1.64 %
7	Inactive	1,007	1.53 %	1,104	1.63 %
		65,650		67,598	

that groups may be classifying their projects strategically (e.g., purposefully misclassifying projects as Beta even though it may be mature for legal or other reasons) or may not have cared about updating the status as the project progressed.

Having said that, the interesting feature of this data is that very few projects (1.64 %) attain mature status. It is most common for projects to be in Beta or less. If this is truly reflective of the state of FLOSS development, motivational theories must incorporate this distribution. At the least, we need an empirical investigation of the characteristics of projects in different development statuses and what that implies.

Diverse Employment Arrangements

A new report by Demos, a UK-based think tank, argues that open-source experts have demolished the differences between categories such as professional and amateur[7]. To emphasize this, consider the story of Jim Hugunin.[8] Jim was well known as an expert in open source. "He created Jpython/Jython; codesigned the AspectJ aspect-oriented-programming language while working at the Xerox PARC research center; and is the moving force behind IronPython, the implementation of the Python language targeted at .Net and Mono." Microsoft lured him to work on their Common Language Runtime team. Jim will continue to work on open source projects while he works for Microsoft. He writes about this transition in this way:

> Over the past year, I've become a reluctant convert to the CLR. My initial plan was to do a little work and then write a short pithy article called, "Why .NET is a terrible platform for dynamic languages." My plans changed when I found the CLR to be an excellent target for the highly dynamic Python language.

Open source developers are expert at breaking walls established by socially-constructed categories. In doing so, they may be rediscovering the meaning of work and establishing new structures. Old labels such as volunteers or employees don't fit many of the complicated work arrangements these individuals find themselves in.

Consider Linus Torvalds. He was a university student when he first created LINUX. Over time, he has been in demand as a speaker, an author, a consultant and a board member. He continues to be involved in the LINUX project, however. This is how he describes his motivation:[9]

> FM: What did you want out of releasing Linux publicly, the first time? Did you get it? Was it money, fame—"reputation"—a nice set of software libraries written by others that helped your other work?

> LT: Originally it wasn't any of the above, although I did ask around for other peoples work that I could use (and thus there was a kind of "quid pro quo" there). Originally

> Linux was just something I had done, and making it available was mostly a "look at what I've done—isn't this neat?" kind of thing. Hoping it would be useful to somebody, but certainly there is some element of "showing off" in there too.
>
> LT: The "fame and reputation" part came later, and never was much of a motivator, although it did of course to some degree enable me to work on it without feeling guilty about neglecting my studies ("Hey, this is much better for me than getting a degree quickly").
>
> LT: A large motivator these days (and this started to happen pretty quickly after making it available) was just that people started using it and it feels good to have done something that other people enjoy using.

Co-Existence of Corporations and Communities

One of the unique properties of the open source software space environment is the co-existence of communities and corporations in competitive and cooperative alignments. Some observers have wondered if this space must be characterized by "altruistic individuals and selfish firms" (Bonccarosi and Rossi 2004). Indeed, firms are selfish and bound by the profit motive and the maximization of shareholder value. Corporations compete in markets and make choices strategically. If they spend any money of a FLOSS product, it would be with the goal of maximizing profits. They are likely to enter into a FLOSS product-market for economic and technological reasons rather than social ones (Bonccarosi and Rossi 2004). Firms routinely use the code of FLOSS products to prepare their own products—e.g., Microsoft's Windows NT software uses code from the popular BSD program with just an acknowledgement in the release notes (Krishnamurthy 2005). At the same time, some corporations have made the code for their products freely available to FLOSS developers and others assign paid employees to work on FLOSS projects.

Corporations provide financial incentives to FLOSS developers in many ways—employment (paid employees participate in FLOSS projects, FLOSS developers are retained as consultants), bounties (financial rewards are provided to developers who can finish a task), sponsorship and grants. In other cases, corporations acquire the rights to a piece of software from a FLOSS group (e.g., Ximian). Corporations are not the only source of financial incentives, however. For instance, the Free Software Directory (<http://directory.fsf.org/>) is supported by a partnership with UNESCO, the Mark Shuttleworth foundation supports many FLOSS projects. Thus, there is great heterogeneity in the type of incentives (magnitude, contingent vs. certain, competitive vs. not) and who provides it (e.g., individual, non-profit organization, for-profit corporation). At this point, the interest has mainly been on the impact of the magnitude of the incentive on motivation and subsequently, behavior.

Co-Existence of the Creative and the Commercial Elements

For some scholars (e.g., Bitzer, Schrettl and Schroder 2004), writing code (or software development in general) is a creative activity. Eric Raymond, an early observer of FLOSS and the author of the popular *The Cathedral and the Bazaar*, has said that:

> It may well turn out that one of the most important effects of open source's success will be to teach us that play is the most economically efficient mode of creative work.[10]

Then, other fields of creative endeavor (e.g., art, advertising, photography) should provide us with a basis for ready comparison. In many creative areas, we see a co-mingling of the commercial and the creative element. A successful artist might paint for the creative and intellectual challenge (i.e., intrinsic motivation) as well as to make a profit (i.e., extrinsic motivation). Those who write advertising copy get paid for their creative work. It would be foolish to question if an artist paints or sculpts to earn money alone. This point is best made through a comment left on two separate discussion boards by zizka.[11]

> Why did Van Gogh paint his paintings? No one ever bought them. He lived off his brother Theo. Theo was an art dealer, fortunately, but unfortunately, he died shortly after Vince. Neither one of them ever profited a dime. What a couple of suckers.

> How do I know that Van Gogh wasn't producing for the market, when he actually was willing to sell painting, and tried to do so? Because he produced painting after painting that wouldn't sell. He did not adapt himself to the market.

Yet, many successful artists earn a lot of money, while managing to be creative individuals pushing the envelope of their oeuvre. Caves (2002) is an interesting examination of creative industries. He identifies several economic principles that are unique to such industries—demand is uncertain, workers care about their products ("artists have strong views") and require diverse skills, products are differentiated and the timing of their creation is often crucial to their success. Many of these principles apply to the area of open source software as well and might help us in building a more general theory of motivation.

A Closer Look at the Evidence

The purpose of this section is to document the important pieces of evidence upon which the academic community's arguments about the motivations of open source developers are based. My goal in chronicling these pieces of data is to refamiliarize readers with them and then use them to make claims about the state of the field. Through a discussion of this data, I will show that the

evidence is much more nuanced than one might expect—most studies find in favor of both intrinsic and extrinsic factors. My goal was to review the important papers—any important omissions are due to error.

Hars and Ou (2001)

This work of Hars and Ou provides some early evidence on developer motivation. They report the findings from an e-mail survey of developers working in open source projects. The developers were identified on the basis of participation in "open source discussion lists and news groups over the Internet." Based on responses from 79 individuals (21% response rate), they provide an analysis of developer motivation and its impact on effort. The data in Table 1 is taken from their paper.

Their analysis provides us with these lessons:

Lesson 1: There are different segments of developers who think and behave differently. In particular, paid developers differ from unpaid developers significantly.

Lesson 2: Both intrinsic and extrinsic motivations are important for all segments. Paid developers ranked some intrinsic motivations highly and unpaid developers ranked some extrinsic motivations highly.

Lesson 3: What ranks high for one segment in terms of motivation may rank low for another segment.

Lesson 4: The motivation that correlates highly with effort for one segment may rank low for another segment.

Even though these findings must be tempered by the sample composition (group membership uncertain) and the response rate, it provides us with some provocative findings about the nature of developer motivation.

Table 1

Hars and Ou's (2001) Analysis of Developer Motivations and Impact on Effort

		All		Students and hobby programmers		Salaried & contract programmers		Programmers paid for OS development	
		Percent	Corr. with effort	Percent	Corr. with effort	Percent	Corr. with effort	Percent	Corr. with effort
1	Internal								
-	Self-determination	79.7%	0.072	81.8%	-0.015	92.6%	-0.303	61.5%	0.221
-	Altruism	16.5%	0.192	24.2%	0.356	11.1%	0.061	7.7%	-0.163
-	Community identification	27.8%	0.116	36.4%	0.361	18.5%	-0.13	30.8%	-0.307
2	External								
2.1	Future rewards								
-	Selling products	13.9%	0.363	6.1%	0.011	3.7%	0.488	53.8%	0.304
-	Human capital	88.3%	0.139	96.9%	0.08	88.5%	0.073	84.6%	0.065
-	Self-marketing	36.7%	0.317	33.3%	0.206	29.6%	0.208	69.2%	0.424
-	Peer recognition	43.0%	-0.021	42.4%	-0.023	48.1%	-0.145	46.2%	-0.178
2.2	Personal need	38.5%	0.304	36.4%	0.301	38.5%	0.186	38.5%	0.328

Ghosh (2002)

This reports results from an extensive survey of 2,784 developers. The top motivations were:

- To learn and develop new skills
- To share knowledge and skills
- Participate in a new form of cooperation
- To improve open source products of other developers
- Think that software should not be a proprietary good
- To solve a problem that could not be solved by proprietary software

The results favor learning, cooperation and problem solving.

Lakhani and Wolf (2005)

This paper presents findings from a survey of developers on projects hosted on Sourceforge (sf.net). Their sample was a 10% random draw from Alpha, Beta, and Production/Stable projects and all Mature projects. Projects with only one developer were systematically excluded. Their survey consisted of 684 individuals from 287 distinct projects making the response rate 34.3%. Thus, this study provides a much larger sample and a controlled research process in comparison to Hars and Ou (2001).

The main findings from this paper[12] are summarized in Table 2. Even though this paper is commonly cited as providing evidence for intrinsic motivation alone, it is striking to see that the results are quite balanced. Both intrinsic and extrinsic motivational factors come out as important.

If one is to consider "need for code" as one variable (i.e., aggregating work and non-work needs), this extrinsic factor emerges as the strongest among surveyed developers. Not surprisingly, paid contributors indicated a signifi-

Table 2

Lakhani and Wolf's (2005) Survey Results

Motivation	Size of effect	% volunteer	% paid	Signi. Diff.?
Enjoyment-based Intrinsic Motivation				
Code for project is intellectually stimulating to write	44.9	46.1	43.1	No
Like working with this development team	20.3	21.5	18.5	No
Economic/Extrinsic based Motivations				
Improve programming skills	41.3	45.8	33.2	Yes
Code needed for user need (work and/or non-work)	58.7	-	-	-
-Work need only	33.8	19.3	55.7	Yes
-Non-Work need only	29.7	37	18.9	Yes
Enhance professional status	17.5	13.9	22.8	No
Obligation/Community based Intrinsic Motivations				
Believe that source code should be open	33.1	34.8	30.6	No
Feel personal obligation to contribute because use F/OSS	28.6	29.6	26.9	No
Dislike proprietary software and want to defeat them	11.3	11.5	11.1	No
Enhance reputation in F/OSS community	11.0	12	9.5	No

cantly higher work-related need while unpaid contributors indicated a significantly higher non-work-related need for code.

The next strongest motivator is the "intellectual need to write code"—an intrinsic factor. Interestingly, there were no differences between paid and unpaid contributors on this variable pointing out the importance of intrinsic motivation *per se*. The third strongest motivator was the need to "improve programming skills." This ranked significantly higher for unpaid programmers—perhaps, a sign that these individuals are not employed. In sum, the top three factors were "need for code" (extrinsic), "intellectual need to write code" (intrinsic) and the need to "improve programming skills" (extrinsic).

Hertel, Niedner & Herrmann (2003)

This paper reports the results from a survey of 141 Linux developers. As before, the main results are summarized in Table 3. Their findings emphasize identification as a Linux developer (perhaps an intrinsic motive) as important. Pragmatic motives (closely correlated with extrinsic motives) were found to be weak predictors of effort, but important predictors of willingness to be associated with the project in the future.

Table 3

Summary of Findings from Hertel, G., Niedner, S. & Herrmann, S. (2003)

	Hours/Week Spent on Linux Development		Willingness to be Involved in the Future	
	Correlation	Signif. Value	Correlation	Signif. Value
Specific Identification (Developer/ Subsystem)	**0.32**	0.01	**0.38**	0.01
Time Loss (Reverse coded)	**0.18**	0.05	0.20	0.05
Pragmatic Motives	**0.16**	0.1	**0.40**	0.01
Norm-oriented Motives	**0.15**	0.1	0.12	0.05
Social / Political Motives	0.01	ns	0.22	0 05
Hedonistic Motives	0.01	ns	0.19	0.05
General Identification as Linux User	**-0.28**	0.05	-0.02	ns

Lakhani and Von Hippel (2003)

This is a study of how Apache users assist each other. Since the focus is on customer service rather than the software development, it is not clear how one should compare this data to previous studies. The main findings from this study is summarized in Table 4. The results are in favor of intrinsic factors (e.g., reciprocity, fun) rather than extrinsic factors (e.g., enhancing future career prospects). In corporations, customer service is usually regarded as a low-status task whereas product development is usually a high-status task. Some of the best engineers in corporations work on developing new products and customer service is frequently outsourced to call-centers. It is not clear if this role hierarchy exists with open source developers as well and if so, how it affects the motivations of those involved. If developers regard coding as a

high status activity, they may be more intrinsically motivated to do that. This study does not provide us with a comparison of motivations to develop software and motivations to provide customer service, however.

Hann, Roberts, Slaughter and Fielding (2002)

This is an interesting study of 325 Apache developers. The authors studied the impact of contribution and rank in Apache to salary and wages. The results are that the magnitude of contributions did not have an effect on salary while rank within Apache had a significant effect.

Roberts, Hann and Slaughter (2004)

This paper extends the survey findings of Hann, Roberts, Slaughter and Fielding (2002) with archival data and provides many provocative findings. The main findings are:

1- The presence of extrinsic motivational factors does not diminish intrinsic motivation.
2- The level of extrinsic motivation affects the level of contributions.
3- The level of intrinsic motivation does not affect the level of contributions.
4- Status-related extrinsic motivations have a stronger impact on contributions.

Von Krogh, Lakhani and Spaeth (2003)

This study presents results from an in-depth study of Freenet. The authors analyze e-mail contributions to Freenet and find some differences between joiners and list participants. The bigger contribution of this paper is the provision of four theoretical ideas—joining script, specialization, contribution barriers, and feature gifts.

These four ideas tie individual motivation with group structure. The idea of a joining script is that motivated individuals cannot necessarily join a group

Table 4

Summary of findings from Lakhani and Von Hippel (2003)

	Frequent providers		Other providers		% Strong
I was motivated to answer because	Mean	Std. Dev.	Mean	Std. Dev.	Agreement
I have been helped before in CIWS-U.so I reciprocate	4.85	2.08	5.14	1.52	48
I have been helped on Usenet before.so I reciprocate	4.61	1.96	5.16	1.53	45
I answer to promote OSS	5.14	1.35	4.76	1.47	33
I answer because its fun	4.81	1 44	4.38	1.49	28
I want to enhance my reputation in OSS/Apache community	4.71	1.35	4 57	1.42	24
I answer to take a break	4.65	1.65	4.22	1.49	20
I answered because I thought the poster would not get a good answer if I did not	4 52	1.57	4.08	1.5	18
I have expertise in this area	4.47	1.32	3.92	1.77	18
I help now so I will be helped in the future	4.52	1.25	5.16	1.38	15
I answer to enhance my career prospects	3.76	1.55	3.57	1.31	6
I answer because it is part of my job	2.23	1 76	2.52	1.75	5
I am the authority in this area	2.47	2.14	2 01	1.56	4

of their choice. In order to join, the group will have to be approached in a manner that is consistent with the group culture. FLOSS developers are seen as specializing in narrow domains. Newcomers tend to specialize that are in need. This specialization will create contribution barriers for some and will allow some newcomers to provide feature gifts to the community.

Summary

The findings of the papers discussed here are representative of an emerging field. We are only beginning to see the evidence about developer motivations. We might conclude the following from the body of research. First, both intrinsic and extrinsic motivational components are important and do exist. Second, the evidence is mixed on the relative value of intrinsic and extrinsic motivational components—notably Lakhani and Wolf (2005) emphasize the place of intrinsic motivations while Roberts, Hann and Slaughter (2004) found no impact of intrinsic motivation on contributions. This should not deter us from exploring the relative value of these components in different contexts. An integrative theory is needed. Third, it is my argument that motivation has been examined by itself and its relationship to other constructs needs to be examined more fully. A few important factors are discussed in the next section.

Moreover, the intrinsic-extrinsic distinction may not be a sufficient taxonomy. Some motivational factors may rightfully claim a place in both categories. For instance, learning of new skills can be an intrinsic motivation since developers may experience the joy of learning. At the same time, this may enhance their future prospect making it an extrinsic motivational factor.

Bonaccorsi and Rossi (2005) have extended the original framework of Feller and Fitzgerald (2002) and placed motivations in three categories—economic, social and technological. The motivations in the economic category correspond closely with extrinsic motivation and the motivations in the social category correlate well with intrinsic motivations. The technological category spans both intrinsic and extrinsic motivations. Combining the two frameworks will provide greater theoretical insight.

Factors That Might Impact Motivation

Four important factors are discussed here—financial incentives, nature of task, group size, group structure.

1. Financial Incentives

Financial incentives play a major role in FLOSS development. As indicated earlier, many FLOSS developers are paid and many companies have made sizeable investments in this area. Yet, we do not see a systematic examination of the role played by these incentives in the way that is outlined here.

To be sure, previous research has mentioned the role of incentives. Yet, except perhaps Hann, Roberts, Slaughter and Fielding (2002), we do not have

an understanding of the impact of financial incentives on individuals, projects, communities and the ecosystem itself. Moreover, Hann et. al. do not make finer distinctions between the types of incentives, when they were provided (i.e., before conducting a task or after) and how they were provided (i.e., direct or contingent). Previous studies (e.g., Hars and Ou 2001, Lakhani and Wolf 2005) have classified developers as paid or unpaid without close attention to how this money was made available to them (e.g., before or after, direct or contingent).

While acknowledging the work of previous authors, it is my argument that in order to gain a fuller understanding of developer motivation what is needed is a more *direct* investigation of the impact of financial incentives on the FLOSS ecosystem. Following previous work on FLOSS as a virtual organization (Marcus, Manville and Agres 2000, Ljungberg 2000), *the ecosystem is defined as a loose collection of individuals and firms working on various projects that lead to the public release of software and source code.* The ecosystem is, thus, at a higher level of abstraction in comparison to a community since (a) it includes individuals and firms and (b) it is not specific to one project.

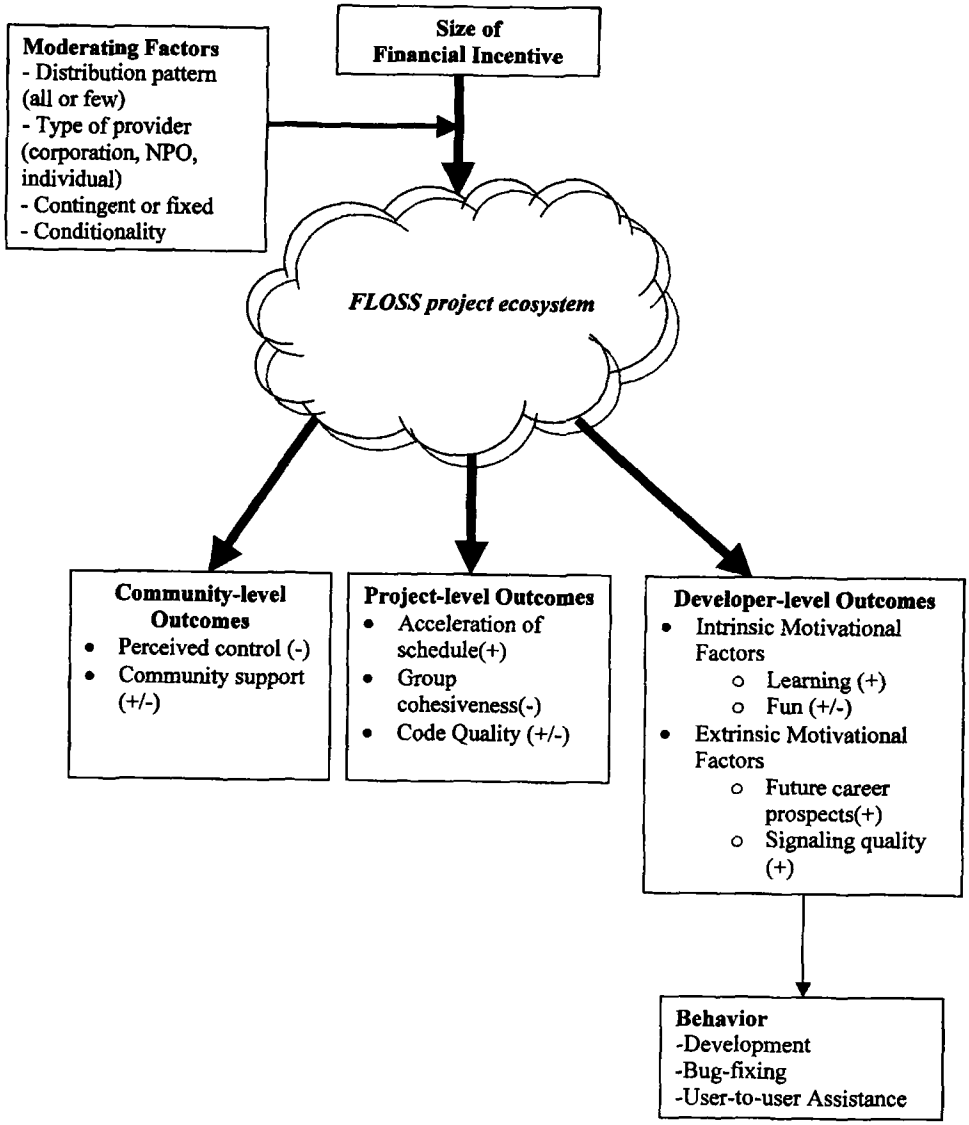
Our interest is at the level of ecosystem because frequently incentives affect developers working on multiple projects. For instance, a bounty may draw from developers from multiple projects rather than from one. Figure 1 is a conceptual framework that depicts the impact of financial incentives on the FLOSS ecosystem. The conceptual framework provides for three types of outcomes as the result of the financial incentive—individual, project and community. Thus, in some cases, there may be deleterious impacts at the individual-level and positive impacts at the project or community level. This helps us flesh out in greater detail the impact of a financial incentive and is hoped that becomes a template for future research.

The point is that not all financial incentives are the same. Regarding all incentives with a certain dollar value to be identical would be misleading at best. In this paper, I identify four variables that help us distinguish between the different types of incentives—Distribution pattern (all or few), Type of provider (corporation, NPO, individual), Contingent or fixed and Conditionality. These are shown in Figure 1. A "+" sign indicates a positive causal relationship and a "-" sign indicates a negative causal relationship.

Financial incentives may be provided to everybody involved in a project or to a few. A corporation hiring away a star developer is an example where the benefit of financial incentives accrue to a few. Thus, a programmer of the stature of Linus Torvalds may get hired easily. But, the common programmer may suffer with few offers. In fact, recent work by economists (Caves 2002) shows that creative industries are characterized by a winner-take-all property where some artists earn a lot of money while most languish with little. If that is the case with FLOSS, we may have a serious sustainability issue at hand small subset of the FLOSS group. On the other hand, if a non-profit or governmental organization provides a grant to a FLOSS project all benefit. When everybody benefits, group cohesiveness is maintained. On the other hand, when financial benefits only to a few, group morale suffers and fragmentation or forking becomes a real issue.

Figure 1

Impact of Financial Incentives on FLOSS ecosystem



Financial incentives may be provided by an individual, a corporation or a non-profit organization. When a corporation provides financial incentives, it increases the chances that an individual involved in the group will get hired. As a result, it provides a chance for more direct compensation. On the other hand, the presence of a corporation is sometimes seen with suspicion in certain communities with a fiercely independent ethos. For instance, when Novell co-sponsored the bounties for Gnome, the discussion boards were abuzz with

theories that Gnome was being asked to implement a Novell to-do list.[13] On the other hand, sponsorship by a non-profit organization (especially those with an international base) may provide a chance for broader distribution and access to the product leading to greater glory to all involved.

Some times, financial incentives to FLOSS developers are contingent on the completion of a piece of software development. Such programs are built on the winner-take-all principle to encourage competition among developers. It is not always rational for an individual developer to enter such a contest (See Appendix I for a formal analysis of the rational response to a bounty). While this could lead to the generation of greater intellectual capital because of the competition, it is likely to break down group cohesiveness with some developers getting the financial reward and not others.

The provision of financial incentives to some developers is conditional on some level of performance. On the other hand, some have been hired unconditionally. Famously, Larry Wall of Perl fame was hired by Oreilly to do "whatever he wants so long as it helps Perl." Others are not afforded the same freedom and may be paid only after they finish a pre-defined job or hired on a part-time basis. Similarly, some are paid after their work is done. A corporation may negotiate the rights to an open-source product after it is released. Individuals may donate to a project to enhance its success after it has achieved some measure of success. On the other hand, incentives to some are not closely tied to having completed a job.

In sum, there is great heterogeneity in the nature of financial incentives. As depicted in Figure 1, the characteristics identified here are seen as moderating the impact of the size of incentive on the FLOSS ecosystem. Some characteristics may strengthen the impact of the incentive while others may weaken. We discuss the moderating impacts after discussing potential outcomes.

As discussed earlier, the outcomes of introducing a financial incentive can be at the level of the individual programmer, the project or the ecosystem. The individual programmer is seen as having intrinsic and extrinsic motivational components with each of these components influencing each other. Extrinsic motivation is expected to increase with the size of the financial incentive—the greater the size of the incentive, the greater the extrinsic motivation.

Developers who work for free on a product may not reject extrinsic financial rewards if they are made available through donations, grants or other mechanisms. Similarly, those who receive financial rewards to participate (e.g., paid employees of corporations) may find participation in a FLOSS project enjoyable and intellectually challenging unlike other environments. Therefore, the impact of the financial incentive on intrinsic motivational factors (e.g., fun, learning) can be understood using the idea of "hygiene factors." Perhaps, while money increases the overall utility of a developer, it does not necessarily increase intrinsic motivation. Therefore, a financial incentive is seen as having a positive effect on intrinsic motivation only under certain conditions.

The total level of motivation within a developer affect behavior—the more motivated the developer, the more the effort towards developing a better product. Intrinsically motivated individuals may be more motivated to perform all

tasks related to a project while extrinsically motivated individuals may be drawn to tasks that lead to greater status. This could be tested empirically.

2. Nature of Task

Up to this point, the discussion of motivation has either explicitly focused on the programmer's motivation to code (or develop software) or has obscured the specific tasks involved. Exceptions include Lakhani and Von Hippel (2003), a study focused on the user-to-user customer service in FLOSS groups rather than the usual focus on software development and the study of bug fixing in papers such as Crowston and Howison (2005).

The development of FLOSS involves many tasks. Just as in for-profit organizations, some of the tasks associated with FLOSS development are more glamorous and come with more status than others.[14] Some scholars have compared the open source software development process to an onion with a core group doing the development and users performing various tasks such as bug fixing at the periphery (e.g., Crowston and Howison 2005, Nakakoji, Yamamoto, Nishinaka, Kishida and Ye 2002). For instance, only the most powerful individuals control which code gets into a release. While fixing bugs is a noble task, it does not have the same cachet as setting the vision for a product's future or writing the specifications for the next round of product development. Newer entrants to the project are frequently assigned to non-glamorous tasks such as bug-fixing while powerful developers work on the code. As a result, we frequently find many individuals at the periphery of the product development process in FLOSS projects (e.g., In a study of Freenet, Krogh, Haefliger and Spaeth 2003 find that the software development process is highly concentrated among a few individuals). Bonaccorsi and Rossi (2003) have noted that "it is difficult to accept the idea that these low gratification activities could be motivated by the same incentive structure than high level, creative work."

While we understand that there is task specialization, we do not know the motivations of individuals to accept less glamorous roles. If the metaphor of a gift economy is to hold, the open source community should have its share of drones willing to do monotonous and boring tasks to move the product further along. What motivates the drones? For instance, do they see bug fixing as the equivalent to peeling potatoes in a restaurant kitchen—a place where you start and work your way up. Alternatively, are they likely to take up this task to reciprocate past contributions from close friends (i.e., a role reversal argument)? Contrasting the motivation of these individuals with those at the core of the process is much needed.

3. Group Size

Despite strong evidence that the vast majority of open source projects are one-person teams (Crowston and Howison 2003, Hunt and Johnson 2002, Healy and Schussman 2004, Madey, Freeh, and Tynan 2002, Madey, Freeh, and Tynan 2004, Hahsler 2004) or "caves" (Krishnamurthy 2002), studies on

developer motivation have systematically excluded these teams (notably, Lakhani and Wolf 2005 and Stewart and Gosain 2004). Others have only investigated large projects—e.g., Lakhani and Von Hippel (2003) investigated Apache and Hertel, Niedner and Herrmann (2003) investigate Linux developers. We have no information to determine the group size of the projects involved in Hars and Ou (2001). Therefore, one might conclude that the most common project form has been excluded from scholarly inquiry about developer motivation! This needs to be corrected in future research.

One might expect that developers associated with one-project teams would be motivated differently from developers in larger teams. One project teams afford the team leader greater control by reduced coordination costs. This may lead to completely different motivation factors from larger groups. At this point, we don't know if that is the case. A study of these groups might provide new leads into what motivates open source developers.

Qualitative insights into the functioning of these groups can be gleaned from a spontaneous Slashdot discussion prompted by Krishnamurthy (2002)[15]. Here are some excerpts:

> I think much of open source software begins as something that gets built by an individual to make his job easier, that he decides to feed back to the community. Generally patches are accepted on merit, but many of the authors are more interested in working on it when convenient than becoming a part time project manager for something which they came up with to same them time in the first place.
>
> [<Source: http://developers.slashdot.org/comments.pl?sid=33715&cid=3645921>]
>
> I've got a couple of GPL'ed projects (Avantslash [fourteenminutes.com], Tellyguide [fourteenminutes.com], MovieGuide [fourteenminutes.com]) and whilst they have a reasonable amount of usage, when it breaks, everyone sits around and waits until I fix it.
>
> So much for the wonderful idea of many eyes helping me out. Maybe with large projects (although I doubt it, tending to think that you only get a small number of people who actually contribute anything) but for the projects like mine it just doesn't happen.
>
> <snip>
>
> I'm fully aware of the Cathedral and Bazaar ideology, but when there is no-one in the Cathedral and you're the only person giving in the Bazaar, GPL suddenly doesn't seem to be this wonderful solution to bugs, features and support.
>
> [<Source: http://developers.slashdot.org/comments.pl?sid=33715&cid=3645974>]
>
> I had an auto mechanic where I used to live who did all the work in his shop, and here's why: Anytime he had a guy around to take some of his workload off, he spent just as much time checking the other guy's work as he would have doing his own. This isn't to

> say the other guy wasn't just as good a mechanic as he was, but he felt that if he was putting his name on the job, he wanted to make damned sure that every bolt involved was done to his high standards. same thing goes for OSS...if you're obsessively writing a piece of code, you might want to make sure it's up to your standards in every way. This doesn't mean others couldn't help, just that due to your own obsessive compulsive nature you feel the need to ensure perfection.
>
> [Source: <http://developers.slashdot.org/comments.pl?sid=33715&threshold=1&commentsort=0&tid=156&tid=8&mode=thread&cid=3646568>]
>
> This is absolutely not startling at all. Most projects are started with an idea. Ideas are generated by individuals. Therefore, most projects are the work of individuals. But the same can be said for "normal" closed-source businesses as well. They are started by individuals.
>
> <snip>
>
> Open source is not really about communities coming together to contribute to a project. Open source is really about communities learning and growing from the shared knowledge of the individuals in that community.
>
> [Source: <http://developers.slashdot.org/comments.pl?sid=33715&threshold=1&commentsort=0&tid=156&tid=8&mode=thread&cid=3645944>]
>
> Think of all the OSS projects and the "names" that go with them — Sendmail, Eric Allman; Samba, Jeremy Allison; Linux, Linus Torvalds; Qmail, Dan Bernstein. Perl, Larry Wall; It goes on and on.
>
> It's actually kind of scary to me since I wonder how many driven-by-one-man projects like this will collapse (i.e., stop development or otherwise rot) without the guy that made the magic originally.
>
> I'm encouraged by the fact that most large, popular projects are too big for one guy to actually do all the coding and usually there are people waiting in the wings who can at least add some legs to these projects.
>
> [Source: <http://developers.slashdot.org/comments.pl?sid=33715&threshold=1&commentsort=0&tid=156&tid=8&mode=thread&pid=3645944#3646898>]

Providing a detailed analysis of these and other comments is outside the scope of this paper. But, the comments are provocative and challenge us to develop a more general theory of developer motivation. See Lin (2005) for a detailed qualitative approach to theory building.

4. Group Structure

Many FLOSS groups have an explicit (e.g., Apache's committee-based structure) or implicit (e.g., Sourceforge projects are controlled by the administrators) organizational structure. Some groups have strong leaders (e.g., Linus Torvalds for Linux). Usually leaders are associated with a group for a long period of time. A FLOSS developer notes that:[16]

> The sense of ownership in Open Source is much more personal. The surest test of this is to look at the longevity of Open Source leaders and the projects they are associated with. Linus Torvalds still leads Linux after nearly fifteen years. Alan Cox is still a key member of the Linux kernel team after a decade or so. Brian Behlendorf still keeps his hand in the Apache project, as he has from the very beginning. Larry Wall is still the chief architect behind Perl, after more than a decade. Eric Allman still guides Sendmail, as he has from the beginning in 1981.

Then, is the motivational structure of a top-notch programmer such as Larry Wall similar to that of a beginner? Perhaps not. A beginner may not have the same access or influence as a Torvalds or a Behlendorf and will have to work his/her way up the food chain. While the person is doing this, what is his/her motivation? Is it similar to a person in a corporation who is assigned low-end work? At least, in corporations, employees are paid a wage. Then, how do we match up motivation with structure? Clearly, members are different points in the structure have different influence.

McGowan (2001) argues that "programmers will be more likely to accept hierarchy in projects that provide unique or unusually valuable returns." In other words, individuals will be happy to act as drones if the project itself is likely to be high impact in nature (e.g., a new operating system). This has not been empirically verified, however.

One of the responses to reduced influence in a large structure may be the formation of "caves" (Krishnamurthy 2002). In other words, some programmers may be trading off being a small fish in a big pond to being the only fish in a small pond. If this is the case, our existing theory needs to be updated to incorporate this.

Conclusion

The issue of employee motivation in the organizational context is old. Corporations want their employees to be maximally motivated and financial incentives are seen as one element of motivation. However, motivation in the context of open source software may be seen as fundamentally different due to the presence of unpaid programmers, implicit rather than explicit forms of control and a different methodology for software development.

The literature in open source has begun to evolve on the lines of intrinsic and extrinsic motivation. It is the hope of this paper that we avoid an either-or stance and develop an integrative theory that acknowledges all elements of

motivation. The four factors identified in this paper—financial incentives, nature of task, group size and group structure—should provide a starting point in this journey.

The consequences of motivation at an aggregate level need to be investigated as well. For instance, there is no empirical evidence on how economic gains are distributed among developers, there is some evidence that prominent developers (e.g., Linus Torvalds, Larry Wall) have captured most of the gains. If that is indeed the case, then that would be a cause of worry for the sustainability of the FLOSS movement.

Above all, the discussion about developer motivation should not break into an either-or debate. Rather, it is my hope that this paper will spur interesting research that studies the inter-relationships between these two components.

APPENDIX 1

Rational Response to Bounty[1]

To analyze this, we take the approach of Selby Jr. and Beranek (1981)—a seminal paper analyzing marketing sweepstakes. Consider a bounty program that announces a bounty, b for the winning entry. Each developer, i, incurs cost of C_i in terms of number of hours spent to prepare the code. Then, following Selby Jr. and Beranek (1981), a risk-neutral participant will enter if-

$$p(b) > C_i \qquad (1)$$

where $p(b)$ is the probability of winning the bounty b. It is likely that those developers whose approaches zero or low levels will find bounty programs interesting. These may refer to developers in these sorts of categories-

- Developers who do not have other opportunities in a recession
- Low-end developers with low skill levels
- Student developers who wish to learn to prepare themselves for market positions
- Developers in countries where wage rates are low

The point made in Selby Jr. and Beranek (2001) about the difficulty of estimating probability ahead of time holds. If a developer does not have a clear understanding of the difficulty of winning the bounty (e.g., estimated number of entries), it becomes hard to evaluate the LHS of equation (1) reducing the likelihood of participation. Observers of marketing sweepstakes and contests have noted that such contests should clearly specify "the specific nature of the prizes available, the exact number of each kind of prize available, and the approximate value of those prizes" in addition to any eligibility limitations (Shapiro 1995). As bounty programs in open source become more prevalent, their regulation will also inevitably increase keeping such considerations in mind.

The analysis so far, however, does not adequately account for the recognition that comes from participating in and winning bounty programs. The utility of a software developer is two-fold if he wins the bounty—the financial amount and a recognition component that comes from the victory. Developers who win bounty programmers are recognized at developer conferences, by providing special mentions on web sites and at community venues. If software developer i values the recognition from winning such programs at , then (1) is modified by

$$p(b + r_i) > C_i \qquad (2)$$

If , we are back to equation (1). If is high, developers will be willing to tolerate low bounty amounts by participating. If is low, developers would only participate at high amounts.

Consider the case when (e.g., a development contest). Then, by equation (2), we see that those developers who have adequately high values will be motivated to enter. Therefore, developers with high will be willing to accept lower bounty payments to make up for their costs.

Notes

1. Quoted in "The Pro-Am Revolution: How enthusiasts are changing our economy and society," Available at- <http://www.demos.co.uk/catalogue/proameconomy/>.
2. I am grateful to Rossi (2004) for bringing this to our attention.
3. This taxonomy is inspired by the work in psychology- notably Deci and colleagues (e.g., Deci, Roestner and Ryan 1999).
4. See <http://www3.ca.com/Solutions/Collateral.asp?CID=61137> for an example of such a bounty.
5. This would be reminiscent of many classic academic disagreements- e.g., nature vs. nurture in psychology.
6. Source: <http://sourceforge.net/softwaremap/trove_list.php?form_cat=6>.
7. Available at- <http://www.demos.co.uk/catalogue/proameconomy/>.
8. Source: <http://www.microsoft-watch.com/article2/0,1995,1654275,00.asp>, Accessed on December 30, 2004.
9. Source: <http://www.firstmonday.org/issues/issue3_3/torvalds/index.html>, Accessed on December 30, 2004.
10. We are grateful to Bitzer, Schrettl and Schroeder (2004), page 9 for alerting us to this quote.
11. Sources: <http://www.j-bradford-delong.net/movable_type/2003_archives/000820.html> <http://dir.salon.com/tech/log/2000/09/21/ultima_volunteers/index.html>, <http://www.invisibleadjunct.com/archives/000134.html>. Zizka is apparently at <http://www.johnjemerson.com/>.
12. This paper also provides the provocative finding that paid developers work longer on all FLOSS projects as well as the focal project. It is not clear why this should be the case.
13. See, <http://mail.gnome.org/archives/foundation-list/2004-August/msg00028.html>, for instance.
14. In corporations, customer service jobs are associated with lower salaries than engineers who develop new products.
15. The entire thread is at- <http://developers.slashdot.org/article.pl?sid=02/06/05/1530208&tid=156&tid=8>.
16. Source: <http://www.oreillynet.com/pub/wlg/4197>, Accessed on December 30, 2004.
17. This analysis also appeared in Krishnamurthy, Sandeep and Arvind Tripathi (2004), "Bounty Programs in Free/Libre/Open Source Software (FLOSS): An Economic Analysis," Under Review for publication in the book-*The Economics of Open Source Software Development.*

References

[Unless otherwise specified, all URLs indicated here were accessible on December 23, 2004.]

Bitzer, J., W. Schrettl & P.J. H. Schröder (2004). Intrinsic Motivation in Open Source Software Development, Unpublished working paper available at- <http://opensource.mit.edu/papers/bitzerschrettlschroder.pdf>.

Bonaccorsi, A. & C. Rossi (2005). Comparing Motivations of Individual Programmers and Firms to Take Part in the Open Source Movement: From Community to Business. Forthcoming in *Research, Technology and Policy.*

Bonaccorsi, A. & C. Rossi (2004). Altruistic individuals, selfish firms? The structure of motivation in Open Source software. *First Monday*, 9(1), URL: <http://firstmonday.org/issues/issue9_1/bonaccorsi/index.html>.

Bonaccorsi, A. & C. Rossi (2003). Why Open Source Software Can Succeed. *Research Policy*, 32(7), 1243-1258.

Caves, R.E. (2002). *Creative Industries: Contracts Between Art and Commerce.* Cambridge, MA: Harvard University Press.

Crowston, K. & J. Howison (2005). The social structure of Free and Open Source software development, *First Monday*, volume 10, number 2, URL: <http://firstmonday.org/issues/issue10_2/crowston/index.html>.

Crowston, K., & Howison, J. (2003). The Social Structure of Open Source Software Development Teams. Working Paper. Available at- <http://floss.syr.edu/tiki-index.php>, Accessed on August 22, 2004.

Deci, E.L, R. Koestner & R.M. Ryan (1999). A Meta-Analytic Review Of Experiments Examining The Effects Of Extrinsic Rewards On Intrinsic Motivation. *Psychological Bulletin*, 125, 627-688.

Feller J. & Fitzgerald B. (2002) *Understanding Open Source Software Development*. Addison Wesley, Boston, MA, USA.

Ghosh, R.A.(2002). FLOSS Final Report of the project "Free/Libre and Open Source Software: Survey and Study." International Institute of Infonomics, University of Maastricht, The Netherlands & Berlecon Research GmbH Berlin, Germany. Available at- <http://www.infonomics.nl/FLOSS/report/>, Accessed on February 14, 2005.

Hahsler, M. (2004). A Quantitative Study of the Adoption of Design Patterns by Open Source Software Developers. *Free/Open Source Software Development*, Editor, S. Koch, Idea Group Publishing, Hershey, PA.

Hann, I.H., J. Roberts, S.A. Slaughter & R. Fielding (2002). Economic Incentives for Participating in Open Source Software Projects. *ICIS Conference Proceedings*. Available at- <http://aisel.isworld.org/pdf.asp?Vpath=ICIS/2002&PDFpath=02CRP33.pdf>.

Hars, A. & S. Ou (2001). Working for Free? – Motivations of Participating in Open Source Projects. *Proceedings of the 34th Hawaii International Conference on System Sciences*. Available at- <http://ieeexplore.ieee.org/iel5/7255/20032/00927045.pdf>.

Healy, K. &A. Schussman (2004). The Ecology of Open Source Software Development. Unpublished working paper. Available at- <http://www.kieranhealy.org/files/drafts/oss-activity.pdf>. Accessed on August 22, 2004.

Hertel, G., Niedner, S. & Herrmann, S. (2003). "Motivation of Software Developers in Open Source Projects: An Internet-Based Survey of Contributors to the Linux Kernel," *Research Policy*, 32, 1159-1177.

Hunt F. & Johnson P. (2002). On the Pareto distribution of Sourceforge projects. *Proceedings of the F/OSS Software Development Workshop*. 122-129, Newcastle, UK.

Krishnamurthy, S.(2002). Cave or Community?: An Empirical Examination of 100 Mature Open Source Projects. *First Monday*, 7(6), Available at- <http://firstmonday.org/issues/issue7_6/krishnamurthy/index.html>.

Krishnamurthy, S.(2005). An Analysis of Open Source Business Models. *Making Sense of the Bazaar: Perspectives on Open Source and Free Software*. Editors- Joseph Feller, Brian Fitzgerald, Scott Hissam and Karim Lakhani, Boston, MA: MIT Press.

Lakhani, K.& E.V. Hippel (2003). How Open Source Software Works: "Free" User-To-User Assistance. *Research Policy*, 32, 923–943.

Lerner, J. & J. Tirole (2004). The Economics of Technology Sharing: Open Source and Beyond. Working Paper, Available at- <http://ssrn.com/abstract=620904>.

Lin, Yuwei (2005). "Hybrid Innovation: The Dynamics of Collaboration Between the FLOSS Community and Corporations," Forthcoming in *Research, Technology and Policy*.

Ljungberg J.(2000). Open Source Movements As A Model For Organising. *European Journal of Information Systems*, 9(4), 208-216.

Madey, G., V. Freeh & R. Tynan (2004). Modeling the F/OSS Community: A Quantitative Investigation. *Free/Open Source Software Development*, Editor, Stephan Koch, Hershey, PA: Idea Group Publishing.

Madey, G., Freeh, V. & Tynan, R. (2002). Understanding OSS as a Self-Organizing Process. *The 2nd Workshop on Open Source Software Engineering at the 24th International Conference on Software Engineering (ICSE2002)*, Orlando, FL.

Markus, L.M., B. Manville & C.E. Agres (2000). What Makes a Virtual Organization Work: Lessons From the Open-Source World. *Sloan Management Review*, 42(1), 13-26.

McGowan, David (2001). The Legal Implications of Open Source Software. *Illinois Law Review* 241(2001).

Nakakoji, K., Y. Yamamoto, Y. Nishinaka, K. Kishida & Y. Ye. (2002). Evolution Patterns of Open-Source Software Systems and Communities," in *Proceedings of International Workshop on Principles of Software Evolution* (IWPSE 2002), 76-85.

Roberts, J., I.H. Hann & S. Slaughter (2004). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. Working Paper, Available at- <http://www-rcf.usc.edu/~hann/publications.html>.

Selby Jr., E. & W. Bernanek (1981). Sweepstakes Contests: Analysis, Strategies, and Survey. *The American Economic Review*, 71(1), 189-195.

Stewart, K.& S. Gosain (2004). The Impact Of Ideology On Effectiveness In Open Source Software Development. Working Paper, Available at- <http://opensource.mit.edu/papers/stewartgosain.pdf>.

von Krogh, G., S. Haefliger & S. Spaeth (2003). Collective Action and Communal Resources in Open Source Software Development: The Case of Freenet. Working Paper, Available at- <http://opensource.mit.edu/papers/vonkroghhaefligerspaeth.pdf>.

von Krogh, G., S. Spaeth & K.R. Lakhani (2003). Community, Joining, And Specialization In Open Source Software Innovation: A Case Study. *Research Policy*, 32(7), 1217-1241.