AN ABSTRACT OF THE THESIS OF

Darren J. Forrest for the degree of Master of Science in Computer Science presented on June 10, 2013.
Title: Corporate Involvement in FOSS Projects: An In-depth Look at Corporate Contributions and Implications for Project Governance

Abstract approved:

____

Carlos Jensen

Free / Open Source Software developers come from a myriad of different backgrounds. While some contribute for personal reasons, many become involved because they receive compensation from corporations or foundations. The motivation for participating in a project can have dramatic impacts on how and what contribution an individual makes. These decisions may align with the needs of the community, the needs of the organization funding the individual, or both. Understanding this dynamic is pivotal to fostering a healthy community. Using socio-technical artifacts of two major FOSS projects we analyze the contributions of corporations and the implications of their involvement. We find evidence that some corporations focus on their own needs with the needs of the greater community an afterthought.

©Copyright by Darren J. Forrest
June 10, 2013
All Rights Reserved

Corporate Involvement in FOSS Projects: An In-depth Look at Corporate Contributions and Implications for Project Governance

by
Darren J. Forrest

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master Science

Presented June 10, 2013
Commencement June 2013

Master of Science thesis of Darren J. Forrest presented on June 10, 2013.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Darren J. Forrest, Author

ACKNOWLEDGMENTS

This thesis would not be possible without the immense help of many individuals. All of the numerous conversations, challenges to my thinking and even emotional support have helped shape myself and this document into what it is today. At the top of the list is my major advisor Dr. Carlos Jensen. Without his constant push to get me to do more and better quality work, what you read today would be a very different experience. His support through hiring me as a TA, encouraging me to travel half way around the world to attend a conference, just to get feedback on my ideas, and occasionally a firm, but kind kick in the pants has been invaluable. The Oregon State University HCI group has provided numerous laughs at the right moment, a sounding board for ideas and a listening ear for frustrations. I am indebted to all of you.

Above all I would like to thank my wife, Launa. Somehow in the heat of deadlines, in the frustration of mistakes she knows how to be the perfect comfort. With kind words, encouragement and support she has helped me more than I could express in the space allowed. Thank you Hon. I love you.

CONTRIBUTION OF AUTHORS

Darren Forrest is the primary author and Dr. Carlos Jensen is the second author of all manuscripts within this thesis. All writing was the work of Darren Forrest.

Data collection for bug repositories was performed by Nitin Mohan and Jennifer Davidsen. Data collection for code repositories and mailing list archives was performed by Darren Forrest. Amir Azarbahkt, Chad Cooper, and Iftekhar Ahmed classified code commits for the Linux kernel.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. Introduction

This thesis contains two conference publications centering on the involvement of corporations in major FOSS projects. The first document was an exploratory study to determine if corporations and other outside organizations were contributing to projects in fundamentally different ways than volunteer contributors. Our hypothesis was that some corporations were contributing in ways focused on their own needs to the detriment of the community. Our findings showed that many corporations were contributing in manners that suggested a self-focused development philosophy, but more work was necessary to resolve some unanswered concerns.

The second publication came out of a desire to answer the remaining concerns from the first study. While the first study showed evidence of corporations contributing in self-serving ways, it did not look for direct evidence of these practices. Additionally, it left out one of the major sources of community involvement. These pieces were considered important to creating a more nuanced picture of how corporations contribute to these FOSS projects and what that would mean for the project's governance. As this second paper is still under review by the conference, references to the first paper are presented in a manner appropriate for double blind review.

Chapter 2. Exploring the role of outside organizations in Free / Open Source Software projects

Darren Forrest, Carlos Jensen, Nitin Mohan, and Jennifer Davidson

Proceedings of the 8th IFIP WG 2.13 International Conference, OSS 2012

Hammamet, Tunisia, September 10-13, 2012

Series vol. 378, no. 1 pp. 201-215.

2.1 Abstract

Free/Open Source Software (FOSS) projects have a reputation for being grass-roots efforts driven by individual contributors volunteering their time and effort. While this may be true for a majority of smaller projects, it is not always the case for large projects. As projects grow in size, importance and complexity, many come to depend on corporations, universities, nongovernmental organizations and governments, for support and contributions, either financially or through seconded staff. As outside organizations get involved in projects, how does this affect their governance, transparency and direction? To study this question we gathered bug reports and commit logs for GCC and the Linux Kernel. We found that outside organizations contribute a majority of code but rarely participate in bug triaging. Therefore their code does not necessarily address the needs of others and may distort governance and direction. We conclude that projects should examine their dependence on outside organizations.

2.2 Introduction

Free/Open Source Software (FOSS) development is a key part of our modern IT infrastructure, responsible for the running of core Internet and server infrastructure. The governance and management of FOSS projects is therefore an essential concern for the continued growth and evolution of the Internet.

FOSS development differs from "traditional" closed-source software in a number of fundamental aspects. One important aspect is that it is not only possible for anyone to view and use FOSS code, but that projects depend on an open participation model where anyone can contribute, and where the best ideas win. This FOSS development ideology is a key strength, as it enables a large and diverse group of developers to pool resources to develop software benefiting everyone.

The culture surrounding FOSS projects can differ substantially, and studies have been done documenting these cultures [16]. In general FOSS projects are seen as meritocracies, where an individual contributors' worth and influence is based upon the quantity and quality of their past contributions to the community. Because of this, despite the fact that FOSS participation is driven by altruism and collaboration [3], there is inherent tension and competition within projects. "Because Apache is a meritocracy, even though all mailing list subscribers can express an opinion by voting, their action may be ignored unless they are recognized as serious contributors" [14].

This inherent competition may be part of the reason why many of FOSS projects are seen as hostile to those trying to join. In a meritocracy, increasing the number of participants means increased competition for resources, or in this case attention and influence. It may therefore be in contributors' interest to erect barriers to ensure fewer people join. Even if one adopts a more benign view of humanity, developers in a meritocracy that primarily rewards code contributions (as is the case with most FOSS projects) are unlikely to "waste" their time writing documentation or mentoring newcomers, as these activities are not rewarded. These factors may in part account for the perceived elitism of some long time FOSS contributors, which can manifest itself in hostility and flaming of newcomers [2, 10].

Another common perception is that FOSS projects are predominantly driven by volunteer efforts. While this was true in the early days, and is still likely true for many smaller projects, studies have shown that a growing number of FOSS developers receive some form of compensation for participation [8]. This compensation can take a number of forms, including release time from other work or monetary or resource donations to fund the work of core project members. This is especially common in larger and more important projects [11].

To a certain extent, compensation is a necessary response to the increased needs of large and important projects. While smaller projects can afford to adopt a more ad-hoc work and leadership model, larger and more crucial projects require more oversight and leadership, something that is difficult to provide with volunteer effort. The fact that an increasing number of FOSS developers are making a living through these projects is a sign of a healthy eco-system. These economic incentives can change the dynamics of FOSS projects. Regardless of whether paid developers are in a leadership position initially, they will tend to drift toward such position because of the meritocracy system. They will be able to dedicate more time to the project, and thus gain more influence.

The distributed organization of FOSS projects and ability for anyone to modify the source code is at the core of what makes FOSS successful. This freedom has to be balanced against the needs of the community, which necessitates cooperation and coordination. The responsibility for managing FOSS projects is in the hands of project maintainers. These individuals manage the code; they are responsible for choosing which contributions to incorporate into a release, and who has the ability to submit code. Because of these powers, they have a measure of control over the direction and participation of the project above and beyond any planning or leadership activities [19, 24].

Control of the code, and thus the direction of a FOSS project, is important. A project may end up alienating, or neglecting the needs of a subset of their users if these are not represented in the project. This is a very real problem. The code-base of the Linux Kernel for instance has ballooned [25] as hardware manufacturers add support for high-performance hardware. While the rapid growth of the code-base may be of only minor concern to those running large data-centers, it can be a serious concern for those wishing to run Linux on minimal hardware.

Despite the importance of code, this is not the only way to contribute to projects. People contribute through bug reporting, documentation, mailing list discussions, mentorship, or governance. It is therefore important to track and understand how participation in these different activities contributes to the health of projects, and the influence different organizations exert through these activities. However, most FOSS projects and researchers focus on only one participation metric. This may lead to a distorted view of what is taking place within their community.

This knowledge is not just important to the projects themselves, but to potential FOSS adopters or developers. Understanding who is supporting and influencing the project is crucial to making better decisions about whether this is a project worth investing in. Having broad support is important; an indicator of the potential and sustainability of a project. The recent and highly public fork of the OpenOffice project should serve as an example of the risks that can be manifest if the direction of a project differs from the desires of the community. The Linux Foundation recognizes the importance of such information in risk analysis and issues a yearly report on its contributor base [15]. Our research may enhance the risk analysis that businesses and other organizations must do by examining the importance of complimentary metrics.

In this exploratory work, we perform a preliminary analysis comparing different metrics tracking participation and influence in projects, whether businesses and other organizations are biased in their participation. To this end we focused on two research questions:

> RQ1: Does bug reporting correlate with code contributions for large organizations?
>
> RQ2: Is there evidence of participation bias, and if so in what direction do organizations tend to lean?

It is important to note that the purpose of our study is not to malign the sponsorship or participation of corporations or governments in FOSS, but to show how these may skew the dynamics of a FOSS project. This influence may not be negative; having professional developers on-board can make a project more successful. However, it is important to be aware of what impact sponsorship can have, and manage the influence that these may have.

In the next section of this paper we review related work. We then discuss our methodology and follow with our key findings, and describe their implications for the future study of FOSS communities and their governance. Given that this is an exploratory study, we follow up with a discussion of the limitations of the study, and important future work. Finally, we wrap up with our conclusions.

2.2 Related Work

There is a growing body of work examining the development practices and governance of FOSS projects [4, 11, 12, 24]. One finding is that FOSS community structure is incredibly diverse. Where one organization might have a well-defined structure of who is doing what, others may operate on a much more ad-hoc fashion.

A number of studies of FOSS communities have relied on bug reporting and code commit records. Ko and Chilana used bug reports to look at how power users impacted the bug reporting process. This can be an especially powerful approach when combined with linguistic analysis of bug reports [12, 13]. Sandusky and Gasser studied bug reports from the Mozilla project to investigate negotiations between reporters and developers [23]. Gall et al studied the evolution of FOSS projects using concurrent versions system (CVS) data for the PACS project [6]. German also used CVS data to study software evolution, but focused on visualization of the development process [7].

To the best of our knowledge no one has used an exhaustive set of project metrics to study FOSS participation. Bug reports, code commits and mailing lists have been used together to explore feature tracking [5], knowledge reuse [17], and the development process [20]. Antoniol et al. sought to connect bug reports with code repository information to allow for easier searching [1]. Each of these combined data from different sources, but did not examine the affiliations of the participants.

Nearest to our work is a series of surveys of FOSS developers and projects (although somewhat dated) [8, 9, 18, 22]. These surveys covered a myriad of topics from demographics to ideology, methodology, and motivations of contributors. Most telling from these studies and further verified by [11] was the employment status of FOSS developers. According to [8], more than 50 percent of contributors are somehow compensated for FOSS development. Jensen found this to be especially true of core developers [11]. Nguyen et al. found that whether bug reporters are paid or voluntary has an effect on the time taken to resolve an issue for some projects [21]. They also found that developers paid to work on FOSS projects were able to resolve more issues because of the increased amount of time those developers had for work on the project.

Most developers work on more than one FOSS project and development is dominated by a few core developers. More than 60% of FOSS participants work on two or more projects [9]. The Orbiten Free Software Survey covered 12,706 developers in 3,149 projects and found that the top 10% of respondents contributed more than 70% of the code. The top ten authors alone contributed almost 20% of all code [8]. This distribution coupled with the meritocracy model suggests that a small number of contributors have very heavy influence over the direction of projects.

According to Bonaccorsi and Rossi, individuals and firms have different motivations for participating in FOSS projects [3]. Firms' motivations for contributing centered on the economic and technological, while individuals were driven by social and personal reasons. Ye and Kishida found that a desire to learn is one of the core motivations for individuals seeking to become involved in FOSS [26]. They also found that community membership and reputation is important to developers.

Joining FOSS projects is not without costs or hurdles [16]. Prospective contributors must familiarize themselves with the constantly changing software as well as any design decisions made or tools used. Von Krogh goes on to say that "the alleged hobbyist culture of open source may not apply at all" [17].

2.3 Methodology

In order to examine our two research questions, even in an exploratory fashion, we needed to carefully narrow our scope. The selection of projects was some concern to the design of the research. We found that many small and medium projects simply did not have enough contributors or sponsors to explore these issues. We therefore restricted our investigation to the Linux Kernel 2.6 and GCC.

We chose these projects because they use complete e-mail addresses in bugzilla and code repositories, data we needed to track contributors. These projects included a diverse enough population that we had a reasonable chance to find and study interesting behaviors. Finally these projects had open and widely available mailing list archives, for future exploration.

To gather data on participation in bug triaging (either as a reporter or as a debugger), we collected the complete bug report and revision history database for each project. We collected and analyzed more than 95% of the bug reports. The remaining bug reports were unavailable due to insufficient permissions, database errors or malformed content.

From these records we extracted the email addresses of anyone who contributed to bug reports. We took the domain from the email addresses and used the publicsuffix 1.0.2 python module[1] to consolidate domains. The crowd-sourced public suffix effort by Mozilla helped us effectively collapse

[1] http://pypi.python.org/pypi/publicsuffix

subdomains such as us.ibm.com and ca.ibm.com to ibm.com. For the purposes of this study we chose not to differentiate between different types of contributors to bug reports. While it is true that those reporting bugs have a different level of influence than those working to fix bugs, they all participate in the public debate about the improvement of the project.

Because we are interested in investigating the influence organizations have on projects, we chose to lump all contributors from an organization together. An organization with a very small number of very active contributors could have more influence than one having a large number of occasional contributors. In order to manage the long tail of occasional contributors, we capped our data such that each domain had to have at least five unique contributors to be included. While it is possible that this could lead to the exclusion of high-volume contributors, it is unlikely that this would affect our understanding of influence and sponsorship.

To make the analysis more meaningful, we grouped organizations together by type: email provider, corporate domain, FOSS project, FOSS umbrella organization, educational institution, government agency, technical association, and unknown. If an email account was provided through some paid relationship or free signup with no other membership requirement, the domain was categorized as an email provider. The same approach applied to domains that were clearly maintained by an individual. FOSS project domains received their own classification while domains that were specifically related to FOSS projects (or FOSS in general), but were not the project itself we categorized separately as a FOSS umbrella organization. Examples of this would be linux.com and gnu.org. Technical associations such as ieee.org and acm.org were categorized separately as well.

For code submissions we gathered the complete commit logs from the projects' code repository. From these we performed the same email parsing and categorization as we did for the bug repository. One central list of domains was used to reduce the risk of incorrect categorization between the two data sources.

Data from bug reports and code repository logs for the Linux Kernel 2.6 was collected from November 6th 2002, through July 29th, 2010. Data for GCC was collected from August 3rd, 1999 through July 30th, 2010.

2.4 Results

If we look at the number of contributors by affiliation, those associated with email provider domains dominate bug reporting (Figure 1), with as many contributors in this category as there are in all the others combined. While the numbers are surprising, it is not an entirely unexpected result, as the barriers to submitting a bug report are generally low, and thus we expected broad participation. Second, a number of paid programmers are likely to not want to disclose their affiliation when reporting bugs in order to protect their employers, deflating the numbers for the other categories.

When we look at code contributions, we see a different trend. Contributors from email provider domains were eclipsed by those from corporate domains. This is also not surprising, since end-users are willing or able to contribute code. Furthermore, contributing code requires a greater time investment; therefore, we expect to see more dedicated, professional programmers. This matches the findings of the Linux Foundation's report that corporations are very active in the coding of the Linux Kernel [15].

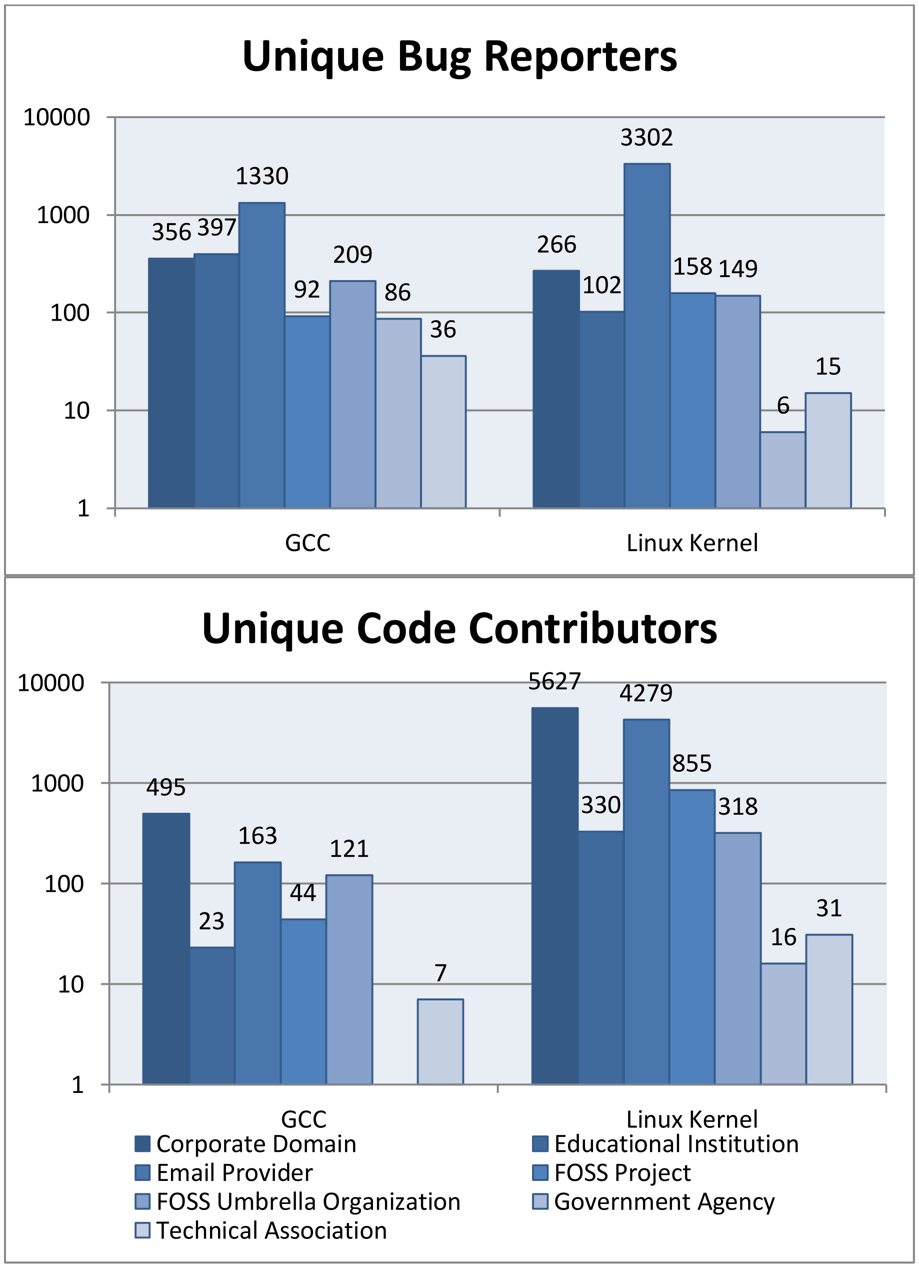

Figure 1 Categories of participation for GCC and Linux Kernel. Logarithmic scale

When we compare bug reporting and code contribution for the Kernel, it is clear that there is a shift in participation, with corporations and other organizations being more involved in coding rather than identifying problems or addressing the complaints of users. Keep in mind that diagrams in Figure 1 are on a logarithmic scale, so seemingly small differences can be very significant.

Another interesting finding is that in the Kernel project there are more unique code contributors from each of the different domain categories than bug reporters. This is somewhat distorted by our filtering of data, but it is still amazing how big the difference there is. Furthermore, because we are only tracking successful code submissions, the number of people trying to contribute code could be even larger. We do not see the same pattern for the GCC project, except for corporate contributors.

So what is going on here? Assuming that bug reporters are not reporting massive numbers of bugs each while code contributors only ever submit one or a handful of code patches, it appears that the Kernel project is driven by a self-centered development philosophy rather than by community needs. By this we mean that people are contributing code because they think the features or improvements will be useful rather than because someone has requested such features or fixes. The discussion about the evolution of the project is not occurring in a public forum.

This analysis however, only scratches the surface. In order to see what goes on, we need to look at individual organizations, and their participation in bug reporting and coding. Again, in order to more clearly see patterns we exclude email providers.

When we examine these tables we find that many top contributors in one column fail to appear in the other (matching pairs highlighted in blue). Only 55% of the organizations with the most code contributors are also in the top 20 in terms of bug reporters/fixers. For the Linux Kernel this drops to 30%.

Unique code contributors		Unique bug reporters	
redhat.com	150	gnu.org	174
gnu.org	104	redhat.com	61
ibm.com	70	ibm.com	55
adacore.com	55	debian.org	46
codesourcery.com	47	sourceforge.net	35
google.com	38	mit.edu	27
apple.com	30	acm.org	26
suse.com	29	intel.com	24
gnat.com	23	hp.com	19
intel.com	17	mpg.de	17
amd.com	14	cmu.edu	16
arm.com	14	berkeley.edu	16
sourceforge.net	12	apple.com	15
debian.org	10	nasa.gov	15
inria.fr	9	utexas.edu	14
ispras.ru	9	cern.ch	14
st.com	8	stanford.edu	13
acm.org	7	suse.com	13
hp.com	7	gentoo.org	13
kpitcummins.com	6	kth.se	12

Table 1 GCC code contribution and bug reporting (top 20 domains)

So what does this mean, and why does it matter? We believe this data shows that some organizations are strategic in how they invest their efforts, choosing to either leverage their strengths (for instance hardware manufacturers like AMD, ARM and TI who have special insight into their own products) or addressing their needs without necessarily contributing to the overall needs of the project (as expressed in the bugs being reported), exemplified here by Google and Novell, among others.

Other organizations choose a different approach, working much closer with the community, regardless of whether they are a hardware provider or services companies. Exemplars here are IBM, Intel, and Redhat, among others, who despite having a vested interest in supporting their own needs balance coding with community engagement. The next step is to see whether participant numbers translate to actual activity, as some organizations can have few people contributing a lot, or a lot of people contributing very little.

Unique code contributors		Unique bug reporters	
ibm.com	721	ibm.com	115
intel.com	571	osdl.org	112
fujitsu.com	478	intel.com	47
redhat.com	409	gentoo.org	36
kernel.org	367	redhat.com	32
google.com	228	sourceforge.net	30
ti.com	209	debian.org	26
sgi.com	203	suse.com	22
linutronix.de	187	hp.com	18
novell.com	145	kernel.org	13
suse.com	132	bigfoot.com	12
amd.com	130	linux.com	12
freescale.com	125	mit.edu	11
nokia.com	104	hut.fi	10
hp.com	96	ubuntu.com	9
atheros.com	89	amd.com	9
samsung.com	88	fujitsu.com	9
infradead.org	83	cornell.edu	8
mvista.com	81	ieee.org	8
oracle.com	78	tudelft.nl	7

Table 2 Linux Kernel code contribution and bug reporting (top 20 domains)

Table 3 shows us that for the GCC project at least, the number of organizations that have more people working on the code rather than contributing and addressing bugs is small, only 8 total. However, if we look at the average number of contributions, we see another source of distortion. Except for the organizations highlighted, the average number of bug contributions per bug reporter is much smaller than the average code contributions per coder. Most organizations may therefore be even more biased toward code contributions than initially thought.

When we turn our attention to the Linux Kernel project we see an even more biased situation. If we rank organizations by the ratio of code contributors to bug reporters/fixers, we find 33 organizations with a code bias, and then a very sharp drop-off. More importantly, the contributions of these code and bug contributors is even more lopsided than in the GCC case, with only the Kernel.org team having bug reporters who are more active than their code contributors.

Domain	Unique Contributors			Contributions per contributor		
	Code	Bug	Ratio	Code	Bugs	Ratio
google.com	38	6	6.333	34.184	2.333	14.652
codesourcery.com	47	8	5.875	43.766	43.875	0.998
redhat.com	150	61	2.459	52.620	15.000	3.508
suse.com	29	13	2.231	221.103	9.077	24.359
apple.com	30	15	2.000	69.300	12.733	5.443
arm.com	14	7	2.000	14.000	1.571	8.912
ibm.com	70	55	1.273	21.314	4.945	4.310
columbia.edu	5	5	1.000	17.200	1.400	12.286
inria.fr	9	11	0.818	4.444	5.273	0.843
st.com	8	10	0.800	12.500	1.900	6.579
intel.com	17	24	0.708	138.765	1.958	70.871
kpitcummins.com	6	10	0.600	1.167	2.200	0.530
gnu.org	104	174	0.598	70.356	145.414	0.484
gentoo.org	5	13	0.385	2.800	3.538	0.791
hp.com	7	19	0.368	25.571	3.947	6.479
sourceforge.net	12	35	0.343	14.500	3.743	3.874
acm.org	7	26	0.269	16.286	2.423	6.721
debian.org	10	46	0.217	19.700	10.478	1.880

Table 3 GCC contributions ordered by coder/bug reporter ratio

Again, what does this mean? It is important to emphasize that there is nothing wrong with organizations contributing large amounts of code; these are very significant contributions. The concern however is that unless these organizations are otherwise engaged in the greater discussion about direction and governance, the contributions may not align with the needs of the project in question. Said another way, if organizations do not get involved in the community discussion (via bug reporting, in this case), they may be effectively ignoring the community.

One very likely situation is that these organizations are responding to bugs reported by their customers directly, or from internal users, circumventing the official bug reporting channels. While this might be understandable from a corporate perspective, this can make it harder to optimally allocate resources, prevent duplication of efforts and make debugging of complex problems difficult for the project overall.

That said, we believe we see clear evidence of corporate strategies with regard to participation on FOSS emerging from our data. For instance, compare the participation of Google and IBM employees across both projects. While IBM does favor code contributions, they still actively participate in bug tracking. We could say that IBM seems to have a balanced approach to participation as the pattern is consistent across the two projects. Google on the other hand seems to consistently follow a very different policy, with very few people reporting bugs, and the bulk of employees focusing exclusively on code. While this could be a coincidence, the pattern seems clear, and it would be surprising to learn that there wasn’t some corporate or incentive policy reinforcing this. Whether that is in the interest of the FOSS projects affected is an open question and one we don’t attempt to answer, but it would likely be in the projects’ interest to be aware of these patterns.

It is entirely possible that some of these organizations have designated email addresses for reporting bugs, or employees dedicated to reporting such issues, thereby skewing our data. This is not entirely far-fetched. Submitting a bug report in the name of a development groups’ email distribution list would ensure that the whole team is notified when someone comments or addresses the issue, as opposed to only the developer who reported the issue. Initial investigations suggest this is not the case, but this is an issue that should be explored in future studies.

Domain	Unique Contributors			Contributions per contributor		
	Code	Bug	Ratio	Code	Bugs	Ratio
fujitsu.com	478	9	53.111	18.866	1.333	14.153
google.com	228	5	45.6	18.741	1.400	13.386
sgi.com	203	7	29	42.291	12.857	3.289
kernel.org	367	13	28.231	41.507	70.692	0.587
amd.com	130	9	14.444	35.400	4.111	8.611
infradead.org	83	6	13.833	140.759	42.333	3.325
oracle.com	78	6	13	184.423	5.833	31.617
redhat.com	409	32	12.781	132.770	5.438	24.415
intel.com	571	47	12.149	79.783	29.319	2.721
vmware.com	43	6	7.167	23.442	2.333	10.048
ibm.com	721	115	6.270	43.431	7.530	5.768
suse.com	132	22	6	391.856	22.500	17.416
hp.com	96	18	5.333	54.448	3.778	14.412
mit.edu	45	11	4.091	44.800	4.455	10.056
linux.org.uk	20	5	4	723.850	87.600	8.263
cam.ac.uk	21	6	3.5	52.476	2.667	19.676
mandriva.com	21	7	3	57.857	1.286	44.990
ubuntu.com	25	9	2.778	16.160	2.222	7.273
acm.org	19	7	2.714	24.053	1.714	14.033
debian.org	67	26	2.577	9.299	3.192	2.913
gnu.org	17	7	2.429	36.588	1.571	23.290
helsinki.fi	13	7	1.857	159.154	1.857	85.705
sourceforge.net	54	30	1.8	21.857	1.000	21.857
cmu.edu	11	7	1.571	10.636	1.571	6.770
ieee.org	12	8	1.5	4.583	1.875	2.444
linux.com	17	12	1.417	90.588	6.750	13.420
gentoo.org	44	36	1.222	63.068	2.722	23.170
berkeley.edu	6	5	1.2	4.333	1.400	3.095
ethz.ch	6	5	1.2	3.833	3.800	1.009
cvut.cz	8	7	1.143	22.500	3.000	7.500
hut.fi	11	10	1.1	14.545	4.800	3.030
uio.no	6	6	1	39.000	28.500	1.368
altlinux.org	6	6	1	9.500	3.333	2.850
tudelft.nl	6	7	0.857	6.167	1.714	3.598
cern.ch	5	6	0.833	2.800	1.333	2.101
osdl.org	27	112	0.241	1193.074	43.795	27.242

Table 4 Linux Kernel code commits to bug reporting ratio

2.5 Limitations

One of the problems we faced in this study was the categorization of contributors by organization type by looking at email domains. This may have led us to misclassify domains, something that would affect the data presented here. While this may have happened, we believe it to be an infrequent occurrence as our categories were relatively well defined.

One place where this may be an issue is in the case of ISPs, where it may be difficult to distinguish the emails of employees from customers. In most cases, additional investigation revealed business rules that dictated which addresses were available to customers and which were available only to employees. Second, participants may be contributing under a generic email address, even if their contribution is part of their work commitment. While we know this occurs, the scope should be limited as most organizations see being involved in FOSS projects as good publicity, or that their name adds extra credibility to their contributions.

A second potential limitation is our decision to exclude any domains with fewer than 5 contributors from the dataset. We did this because of the sheer number of domains we needed to categorize. By applying this filter we were left with some 500 domains from over 13,000 original domains. While we may have lost some high-impact contributors, our goal was to determine the impact of organizational, rather than individual, participation in FOSS projects. Given that these were very large projects, we feel that an entity dedicating so few resources out of the project total is unlikely to have that much influence. There will always be exceptions, but we believe the overall impact of this decision is negligible.

The projects included varying information in the CVS data. For example, the Linux Kernel has a very structured format for their code commits. Each code commit has an author as well as a list of additional individuals who sign-off, review, or are otherwise included in the commit log. GCC does not follow as rigorous of a process. This difference in practices could have had an effect on our results, with contributors being over or undercounted.

Finally, our analysis of contributions, both to the debate as well as to the code base was very simplistic; a simple count. We acknowledge the fact that not all code contributions or bug report interactions are created equal, some of these will be more important than others. A simple count gives a distorted view. However, without a rating or review system for contributions, we have no objective way of evaluating the impact of individual contributions.

2.6 Conclusions

We found that for these large projects, corporate developers dominate in terms of code contributions. This has important implications for project governance and our understanding of FOSS demographics. Large projects may not be accurately portrayed as grass-roots volunteer efforts.

The data suggests there exist two distinct communities within projects. While these communities may interact with each other through other means (e.g. mailing lists), there is a community of coders and a community of bug reporters. While this is not unexpected, it is unexpected to see that the most prolific code contributors seem not to interact with the bug reporters—we tracked any participation in bug reporting, not just the reporting of new bugs.

This disconnect can in the long-term lead to alienation and declining participation of non-technical contributors.

We also found that many projects do not currently track this kind of data, or at least they do not make it publicly available. While there may be privacy concerns with posting email addresses or calling out individual developers or companies, this has to be balanced against users and other contributors' need to know. Without this information, FOSS users and possible contributors lack the necessary information to understand whether a project is well governed and healthy.

2.7 Future Work

In the future we plan to expand our scope both in terms of projects examined and metrics used. For instance, we hope to look at projects that range in size. Prior research has shown that projects studied are anomalies rather than the norm in the FOSS ecosystem. Examining how smaller projects are affected would give us a better picture and help their maintainers make better growth decisions.

Our research primarily used publicly available data. While this is important for evaluating the transparency and inclusiveness of decision-making, we know we are missing part of the picture, including any private deliberations between maintainers. We hope to get the direct cooperation of projects to determine if understanding participation in FOSS projects differs with an inside view.

The involvement of government agencies warrants further investigation, as we believe that these agencies have much to offer the FOSS community. We wish to explore how these organizations contribute, and how to get them more involved.

Recent events in the OpenOffice/LibreOffice project have brought the issue of forking and the role of corporations in FOSS to the forefront. We plan to investigate these projects as well as others that have forked over governance issues to determine if our metrics are meaningful. Retrospective analysis, before and after the split, could give key insights and early warning signs to enable corrective actions if desired.

Bug reports and code commits are not the only means by which individuals are involved in FOSS development. In the future we plan to look at mailing lists, project governance, project documentation, and conduct developer interviews. These will give us a broader picture of FOSS development work. This may help in answering more difficult questions relating to measuring project health and success. We hope to better understand healthy participation.

2.8 Acknowledgments

We would like to thank Andy Ko for his assistance with scripts to extract bug repository information from Bugzilla databases. We would also like to thank the Oregon State University HCI group for their input and feedback on the research.

2.9 References

[1] Antoniol, G., Gall, H., Di Penta, M., & Pinzger, M. (n.d.). Mozilla: Closing the Circle. Retrieved from http://scholar.googleusercontent.com/scholar?q=cache:neQwzT9UfPwJ:scholar.google.com/&hl=en&as_sdt=0,38

[2] Bergquist, M., & Ljungberg, J. (2001). The power of gifts: organizing social relationships in open source communities. *Information Systems Journal, 11*(4), 305–320.

[3] Bonaccorsi, A., & Rossi, C. (2003). Why Open Source software can succeed. *Research Policy, 32*(7), 1243-1258. doi:10.1016/S0048-7333(03)00051-9

[4] Crowston, K., Annabi, H., Howison, J., & Masango, C. (2004). Effective work practices for software engineering: free/libre open source software development. *Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research*, WISER '04 (pp. 18–26). New York, NY, USA: ACM. doi:10.1145/1029997.1030003

[5] Fischer, M., Pinzger, M., & Gall, H. (2003). Analyzing and Relating Bug Report Data for Feature Tracking. *Reverse Engineering, Working Conference on* (p. 90). Los Alamitos, CA, USA: IEEE Computer Society. doi:http://doi.ieeecomputersociety.org/10.1109/WCRE.2003.1287240

[6] Gall, H., Jazayeri, M., & Krajewski, J. (2003). CVS release history data for detecting logical couplings. *Sixth International Workshop on Principles of Software Evolution, 2003. Proceedings* (pp. 13- 23). Presented at the Sixth International Workshop on Principles of Software Evolution, 2003. Proceedings, IEEE. doi:10.1109/IWPSE.2003.1231205

[7] German, D. M. (2006). An empirical study of fine-grained software modifications. *Empirical Software Engineering*, *11*, 369-393. doi:10.1007/s10664-006-9004-6

[8] Ghosh, R.A., & Prakash, V.V. The Orbiten Free Software Survey. First Monday, 5(7), July 2000, http://www.firstmonday.org/issues/issue5_7/ghosh

[9] Hars, A., & Ou, S. (2001). Working for Free? - Motivations of Participating in Open Source Projects. *Hawaii International Conference on System Sciences* (Vol. 7, p. 7014). Los Alamitos, CA, USA: IEEE Computer Society. doi:http://doi.ieeecomputersociety.org/10.1109/HICSS.2001.927045

[10] Jensen, C., King, S., Kuechler, V. (2011). Joining Free/Open Source Software Communities: An Analysis of Newbies' First Interactions on Project Mailing Lists. *44th Hawaii International Conference on System Sciences*

[11] Jensen, C., & Scacchi, W. (2007). Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study. *29th International Conference on Software Engineering, 2007. ICSE 2007* (pp. 364-374). Presented at the 29th International Conference on Software Engineering, 2007. ICSE 2007, IEEE. doi:10.1109/ICSE.2007.74

[12] Ko, A. J., & Chilana, P. K. (2010). How power users help and hinder open bug reporting. *Proceedings of the 28th international conference on Human factors in computing systems,* CHI '10 (pp. 1665–1674). New York, NY, USA: ACM. doi:10.1145/1753326.1753576

[13] Ko, A. J., Myers, B. A., & Duen Horng Chau. (2006). A Linguistic Analysis of How People Describe Software Problems. *IEEE Symposium on Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006* (pp. 127-134). IEEE. doi:10.1109/VLHCC.2006.3

[14] Kogut, B., & Metiu, A. (2001). Open-Source Software Development and Distributed Innovation. *Oxford Review of Economic Policy, 17*(2), 248 -264. doi:10.1093/oxrep/17.2.248

[15] Kroah-Hartman, G., Corbet, J., & McPherson, A. (2008). Linux kernel development: How fast it is going, who is doing it, what they are doing, and who is sponsoring it. *The Linux Foundation Publication* http://www. linuxfoundation. org/publications/linuxkerneldevelopment. php.

[16] Krogh, G. von, Spaeth, S., & Lakhani, K. R. Community, joining, and specialization in open source software innovation: a case study. *Research Policy* (Vol. 32, 7). Open Source Software Development Jul 2003, pp. 1217-1241.

[17] Krogh, G. von, Spaeth, S., & Haefliger, S. (2005). Knowledge Reuse in Open Source Software: An Exploratory Study of 15 Open Source Projects. *Hawaii International Conference on System Sciences* (Vol. 7, p. 198b). Los Alamitos, CA, USA: IEEE Computer Society. doi:http://doi.ieeecomputersociety.org/10.1109/HICSS.2005.378

[18] Lakhani, K., & Wolf, R. G. (2003). Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. *SSRN eLibrary*. Retrieved from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=443040

[19] Lessig, L. (1999). *CODE and other laws of cyberspace*. Basic Books.

[20] Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol., 11*(3), 309–346. doi:10.1145/567793.567795.

[21] A. Nguyen Duc, D. S. Cruzes, C. Ayala, and R. Conradi, "Impact of Stakeholder Type and Collaboration on Issue Resolution Time in OSS Projects," in *Open Source Systems: Grounding Research*, vol. 365, S. A. Hissam,

B. Russo, M. G. Mendonça Neto, and F. Kon, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1-16.

[22] P. David, A. Watermann and S. Arora, "FLOSS-US: The Free/Libre/Open Source Software Survey for 2003." *Technical Report*. Stanford Institute for Economic and Policy Research, Stanford, USA, 2003. http://www.stanford.edu/group/floss-us/

[23] Sandusky, R. J., & Gasser, L. (2005). Negotiation and the coordination of information and activity in distributed software problem management. *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work,* GROUP '05 (pp. 187–196). New York, NY, USA: ACM. doi:10.1145/1099203.1099238

[24] Scacchi, W. (2004). Free and open source development practices in the game community. *IEEE Software, 21*(1), 59- 66. doi:10.1109/MS.2004.1259221

[25] System Size - eLinux.org. (n.d.). Retrieved October 8, 2011, from http://elinux.org/System_Size

[26] Ye, Y., & Kishida, K. (2003). Toward an understanding of the motivation of open source software developers (pp. 419-429). IEEE. doi:10.1109/ICSE.2003.1201220

Chapter 3. Corporate Influence in the Linux Kernel

Darren Forrest and Carlos Jensen

Submitted to CSCW 2014

3.1 Abstract

Free / Open Source Software developers come from a myriad of different backgrounds. While some contribute for personal reasons, many become involved because they receive compensation from corporations or foundations. The motivation for participating in a project can have dramatic impacts on how and what contribution an individual makes. These decisions may align with the needs of the community, the needs of the organization funding the individual, or both. Understanding this dynamic is pivotal to fostering a healthy community. We build upon the work of Forrest et al. in this multiple case study of corporations contributing to the Linux Kernel. We find evidence that some corporations focus on their own needs with the needs of the greater community an afterthought.

3.2 Introduction

Free/Open Source Software (FOSS) is a key component of our current computing ecosystem. FOSS projects are foundational in multiple arenas of the Internet as well as internal systems used by corporations around the world. FOSS is even commonly found on personal computers. The success and growth of FOSS projects is important to the overall future of computing. Therefore, the study of what makes FOSS projects successful deserves a prominent place in research.

The development of FOSS is different from traditional software development in corporations. In traditional software development, access to the source code is limited to the development team, whereas in FOSS access is freely available to all. In addition, most projects encourage open participation. This means that anyone with the desire and expertise necessary to contribute can get involved with and contribute to the project. These fluid boundaries, and what they mean for team management, makes the community structure and process of FOSS significantly different than that of traditional software teams.

With free access to the source and open participation, some control must be exerted to prevent chaos. Central to most FOSS projects is the meritocratic system, where contributions and contributors are judged by what they contribute to the project. Project maintainers hold power by deciding which submissions are included and which are rejected [9]. "Because Apache is a meritocracy, even though all mailing list subscribers can express an opinion by voting, their action may be ignored unless they are recognized as serious contributors" [25]. This process is successful in most circumstances, but as De Souza points out in regard to code submissions, FOSS rhetoric focuses on openness and access, but FOSS in practice focuses on regulation and control [9].

In spite of this control structure, FOSS development is often considered a grassroots movement. Developers are said to contribute because of altruistic motives [5]. While this may be true of some developers, recent studies have shown this perception is not always accurate [18]. Compensation is fairly common among developers of FOSS, at least for the larger projects [21]. Some developers are paid directly by sponsors for their work on FOSS, while others receive funding through a foundation or other non-profit organization [16]. Still others receive compensation in the form of equipment or time off work for FOSS development. All of these forms of compensation make FOSS development a complicated economy.

The concept of a grassroots organization is also inherently in conflict with the meritocratic model. Grassroots organizations thrive on an influx of new and willing members. Within a meritocracy, individuals are in a constant struggle for the attention of decision makers or for resources that will help them in their work. In this way it can be to an individual's best interest to prevent the inclusion of other members who would also compete for those resources or attention. This may be one of the reasons for the flaming and elitist tendencies the Linux Kernel has become known for [2, 22].

Large projects, by necessity, have more formalized processes. As a result, one would expect to see more compensated developers in core roles as the demands of the task requires more effort and consistency than what can be supported through volunteer labor. However, mixing volunteer labor with paid labor may change the dynamics of these projects. It is not much of a stretch to assume paid developers could be biased to pay extra attention to the needs of the organization they represent, to the detriment of the needs of the rest of the community. Additionally, because of the meritocratic system and the additional time paid developers are able to devote to the project, they and their contributions will likely be valued more highly. Because of this, it is important to determine if individuals who are compensated for their work on FOSS projects have disproportionate influence over a project than volunteers.

We therefore, seek to understand the prevalence of individuals aligned with corporations and how this might affect the social dynamics of FOSS projects, and the projects' evolution. Forrest et.al began this investigation by looking at the Linux Kernel and GCC projects [14]. They surveyed bug and code repositories to determine the affiliations of contributors. That exploratory work provides some interesting trends but raises additional questions that need investigation. Our work seeks to answer some of those unanswered questions.

Forrest et al. found that corporations had substantially larger numbers of developers making contributions to the code repositories than were working in the bug repository. In other words, they found it likely that corporations were more likely to sponsor new code rather than engage in community-driven initiatives. From this they drew the conclusion that corporations had a more self-focused development philosophy rather than one centered on the needs of the community. This analysis looked at the number of developers and contributions in the two repositories without taking into account the mailing list as a forum for discussing community needs or a qualitative analysis of the contributions.

To expand on the work of Forrest et al. and provide a more nuanced view of the manner that corporations involve themselves in FOSS projects we put forth the following research questions:

RQ1: Does mailing list activity show a lack of corporate participation in community discussion?

RQ2: Do code contributions show corporations are contributing more new features or fixing bugs?

It is important to note here that the purpose of this work is not to malign or downplay the good that corporations do for FOSS projects. FOSS projects depend on the contributions of all their participants and firms are an integral part of that process. Instead our goal is to determine how firms are participating and how that participation impacts the project as a whole. This knowledge will allow project maintainers to better understand the dynamics of sponsorship, how to maintain direction and how to foster teamwork in the complex environment. The outline of this paper is as follows. We discuss related work in the FOSS space and the techniques used. After that we describe our research methods, followed by the results and discussion. Finally we provide conclusions and outline future work.

3.3 Related Work

FOSS projects have been well studied over the past decade. During that time researchers have advanced many theories about community development [6], software development processes [34], and even the quality of the software itself [17].

FOSS community structures have been studied and several models have been proposed. Perhaps the best known is the onion model [7, 10, 11, 19, 21, 27, 28, 33, 37, 38]. The onion model describes a set of roles and role transitions seen within FOSS, with the more specialized and greater authority roles being occupied by more senior or capable members of the community. Individuals progress through these roles from the outer, less important roles to the inner, more important ones.

Jergensen et al. expanded on this work by looking at how developers migrate between groups and proposed the idea of an onion patch [23]. In this model experienced developers are able to move more quickly through the ranks of developers or skip levels entirely because of their experience and reputation with other FOSS projects. Herraiz et al found that developers who are paid to contribute to FOSS have a “sudden integration” process, effectively skipping levels, in contrast to volunteer developers who more closely follow the onion model [19]. Still others have suggested that the onion model is only accurate for a handful of projects and should not be considered true of FOSS in general [7].

Joining a FOSS project is not without cost. Krogh et al. found that joining a project requires a significant investment of time and effort as well as expertise [37]. A high cost of contributing may lead to higher developer turnover. Methods of measuring developer turnover was studied by Robles [31]. Retention of developers has also been studied in the hopes of determining factors that will contribute to the success of FOSS [35].

FOSS work often takes place in a distributed environment, through mainly text-based communication. The artifacts of this communication have been used in numerous studies of FOSS development. Some of the major sources of this communication are bug reports, code repositories, and mailing lists.

Bug reports are vital to any FOSS project. The need for open bug reporting and many users reporting bugs is well recognized by Linus’ law, which states that “many eyes make all bugs shallow” [30]. Bettenburg found the quality of bug reports important enough to study the features of good reports and provide tools to help users provide better reports [4]. Bug reports are one of the ways end users interact with the development team [32]. Bug reporting is even used by some developers to prioritize development work based on the number of duplicate bug reports [3]. While bug reports are almost always considered a positive feature of FOSS development, Ko and Chilana found that power users reporting bugs can have a negative impact on the project [24].

Concurrent Versions Systems (CVS) are where projects control and manage the code that is accepted. German focused on ways to visualize code changes [15]. Analyzing source code data has even been used to study things like developer turnover and role identification within projects [31]. De Souza et al found that the structure of the code mimicked the structure of the development team [9]. More importantly for our work, they found that control over the code was managed through the CVS, and that core development teams used this to uphold their decisions within the project and community.

Mailing lists are the forum used by most FOSS projects to make community decisions [28]. Mailing lists are used to broadcast messages to all the members who have subscribed. In a similar manner to the CVS, the mailing list can be a forum of controlling the behavior of participants [9]. Conflict and confrontations in mailing lists have been studied by Jensen et al [22] and Bergquist and Ljungberg [2]. For the Linux kernel, the Linux Kernel Mailing List (LKML) is seen as the primary forum of discussion and decision making [20].

Several surveys of FOSS developers have been undertaken in the last ten years [8, 16, 18, 26]. While we have some concerns over the distribution of the respondents, these surveys are the best information available at this time. The survey topics included questions concerning demographics, development methodology, motivations and ideology. Most important in this work were questions regarding compensation and employment status of the developers. While the findings on compensation rates varied widely, Hars and Ou claim 16% FOSS developers are directly paid for FOSS work with an additional 34% considering FOSS development part of their job expectations [18]. Core developers had higher rates of compensation according to Jensen and Scacchi [21].

Further work by Nguyen found that compensation had an effect on work efficiency in FOSS projects [29]. Nguyen studied bug report resolution time and claim that developers who are paid for the FOSS work are able to resolve more issues faster than their non-paid counterparts in some of the projects studied. Naturally those developers were able to resolve a greater number of issues because of the increased amount of time they were able to spend. This finding combined with the meritocratic model of FOSS supports our assumption of greater numbers of paid developers with core roles in a project.

Compensation is just one reason for participating in FOSS; research has also been directed at other motivations for participating in FOSS [26]. At the macro level, Bonaccorsi and Rossi surveyed corporate leaders and found that firms and individuals had different motivations for participating in FOSS [5]. Corporations tended to focus on technical and economic reasons, while individual respondents claimed social and personal reasons. Ye Kishida found community membership, a desire to learn, and reputation were important to individuals [38].

Conflict is a common part of any group work, especially when differing motivations or values are involved. The distributed work style of FOSS projects also presents its own set of challenges, which have also been studied [12]. Within the realm of FOSS, mitigation of conflict has been the subject of some work. Elliot and Scacchi found that the social ties within a development community are one of the key ways to mitigate conflict [13]. Stewart and Gosain identified and studied several core FOSS values and how they impacted the success of FOSS projects [36]. They found that affective trust was one of the main drivers for successful FOSS projects. While there are other ways of mitigating conflict and supporting a project, making and maintaining the social ties and fostering trust are wise actions for anyone seeking success in FOSS.

3.4 Methodology

Because we extend the work of Forrest et al [14], our methodology closely mirrors theirs. We chose to look in more detail at the Linux kernel 2.6, the most interesting project from their paper. While the work of Forrest et al. [14] was based exclusively on data from bug and code repositories, we provide a more nuanced and complete picture of corporate participation by looking at bug and code repositories as well as mailing lists.

We used a multiple case study methodology looking at the eight most prominent corporate participants, as measured by number of code contributors, and their involvement in kernel development. Our selection includes both hardware and software companies.

Hardware companies are naturally expected to be heavily involved in coding, as they add drivers and support to their products, potentially at the expense of other community participation. Because these features require detailed working knowledge of the hardware, and possibly access to proprietary information, it is natural to see them perform this work. We selected two companies to study that were highly skewed toward code submission, Fujitsu and Samsung, and two that have a more balanced approach, AMD and Intel. We followed a similar selection process for software companies, selecting Oracle and Google for code heavy participation and Redhat and IBM for the more balanced approach. Many organizations use subdomains. To collapse these we used the publicsuffix 1.0.2[2] python module.

To answer our research questions we used scripts to gather 95% of the bug reports for the Linux kernel. The remaining reports were inaccessible either due to insufficient permissions or other repository errors. Our data included the full email address of each reporter, assignee, and commenter for a bug. We filtered to focus on the corporations of interest.

Data for code submissions was similarly gathered. We downloaded the Linux kernel source and used git log information to obtain author names, email addresses and commits. We then filtered our data to focus on the corporations of interest.

We gathered mailing list data from the University of Indiana LKML mirror site[3]. We used this mirror because it allowed grouping of messages based on authorship. Since LKML sites do not include the author's email address, drawing connections between code, bug, and mailing list contributors required the matching of real names to the email addresses gathered from the bug and git repositories. We successfully mapped 98% of email addresses involved in the bug repository to a name used in the LKML.

With code submissions we had significantly less success even though we used the ".mailmap" git log options to catch typos and multiple spellings associated with a single address. With the code repository data we had to account for several unusual characteristics. 40% of the email addresses were obfuscated in some manner. Examples took the form of:

---

[2] https://pypi.python.org/pypi/publicsuffix/

[3] http://lkml.indiana.edu/hypermail/linux/kernel/index.html

CD45F355109A9B@domain.com, which appeared at first to be commit hashes. Further investigation revealed these were reference addresses to specific LKML messages related to the code submission. We chose to exclude these addresses from our dataset because they would not be tied to any one individual. After removing these addresses we were able to match 98.5% of email addresses with real names. Based on real names, 115 individuals appeared to use multiple addresses. These addresses were consolidated to single entities except when changes involved multiple corporations.

We then analyzed the data from all three sources by looking at the number of committers from each organization, the number of commits per individual, and the overlap in committers between the three data sets.

To categorize code commits we considered the types of commits that corporations may make to the kernel: bug fixes and improvements, or new features or functionality. We labeled commits made in response to a specific or implied bug report as bug fixes. Keywords included: Fix, Bug, and Resolves along with their derivatives. Improvements were identified based on the keywords: Cleanup, Optimize, and Simplify or their derivatives. Commits were tagged as New Features by the keywords: Add, or Introduce. Also the number of lines modified was compared with the lines added. Those commits with greater numbers of lines added were considered more likely to be new features. Anything that did not fit into this pattern we marked as unclassifiable. 10.3% of the commits investigated fell into this latter category.

We randomly selected 50 commits from each of the 8 corporations and categorized those as a representative set of the work the corporation contributed to the Kernel. Two evaluators worked independently to classify the commits.

Their datasets had a 33% overlap, which we used to calculate the inter-rater reliability, which gave us a Cohen's Kappa of 0.49. Bakeman et al. highlight two factors, which our dataset has, that make lower kappa values acceptable: few codes and codes that are not equiprobable [1]. In simulations, assuming equiprobable codes, raters with 85% agreement would have 0.49 and 0.60 kappa values for code sets of size 2 and 3 respectively.

Data for the Linux kernel was gathered for dates beginning the 6th of November 2002 and until the 29th of July 2010. This gave us nearly 8 years of history to analyze.

3.5 Results

One of the concerns in the work of Forrest et al. is the use of the number of contributors in place of the number of contributions [14]. We investigated this concern and found that while the number of contributors may not be directly interchangeable with the number of contributions, they do follow the same trends of much greater numbers of code contributions than bug contributions for each of the corporations studied (Table 5).

After tallying the data we see numbers similar to those of Forrest et al as shown in Table 5. Our numbers are slightly lower than Forrest et al because of removing obfuscated email addresses, but the same trends are still seen.

The data is also interesting from the perspective of who is participating in more than one aspect of the Kernel community. We can see from figure 1 that most of our corporate participants are contributing in the git repository. It is important to note that since we only had access to the names given in the LKML, we could not determine if an individual belonged to one of these eight companies unless they were participating in the bug or code repository as well. Because of this, we have not included a LKML only category.

Company	Code		Bug		LKML	
	Contributions	Contributors	Contributions	Contributors	Contributions	Contributors
AMD	3360	43	37	9	3189	11
Fujitsu	6272	58	13	10	1086	16
Google	2316	83	7	5	13239	35
IBM	29567	346	866	112	22611	173
Intel	42854	256	1378	47	18570	61
Oracle	14942	35	35	6	11804	17
Redhat	48052	213	174	31	40285	116
Samsung	1200	34	1	1	56	3

Table 5 Contributors and contributions by company

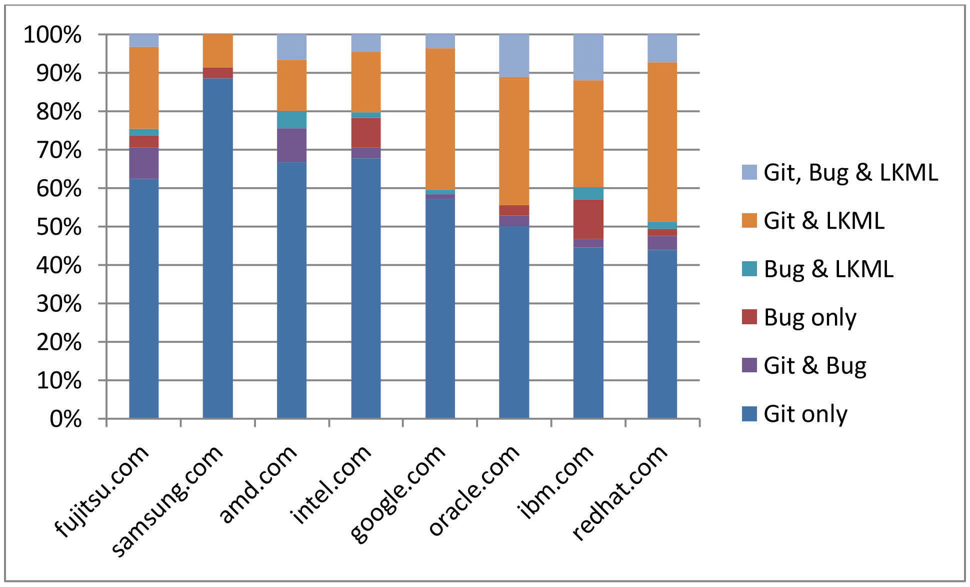

Figure 2 Distribution of contributors between bug, code and mailing list repositories.

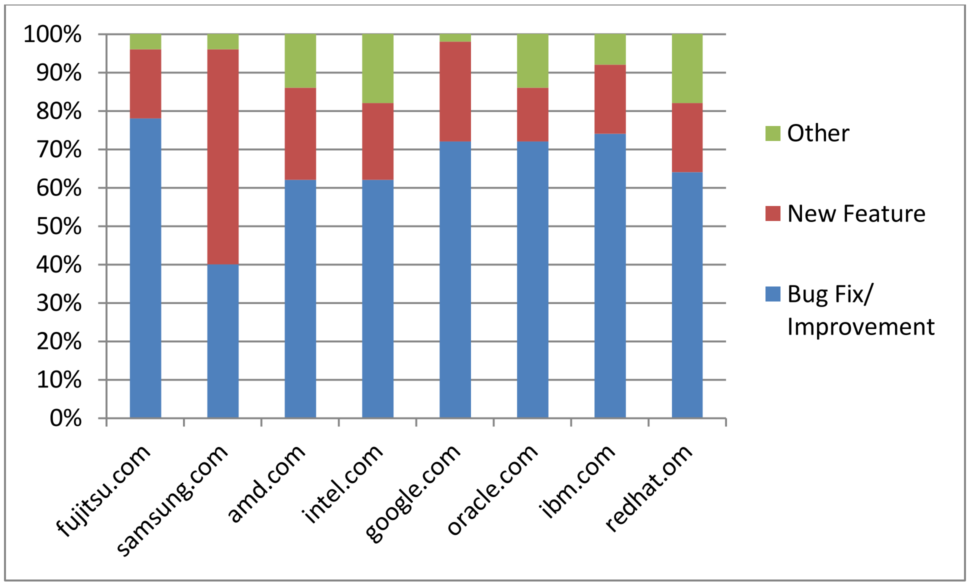

Figure 3 Contribution type by company.

Company	Forrest et al	Our Data
amd.com	14.4	4. 8
fujitsu.com	53.1	5.8
google.com	45.6	16.6
ibm.com	6.3	3.1
intel.com	12.1	5.4
oracle.com	13	5.8
redhat.com	12.8	6. 9
samsung.com	88	34

Table 6. Ratio of code contributors to bug contributors

The goal of classifying code commits was to reveal a better picture of how corporations are contributing code to the Linux kernel. In Figure 3 we can see that a majority of corporate code commits are an improvement or bug fix of some kind. There is a slight difference in the bug fix / improvement to new feature ratio between software and hardware companies. Most striking is the focus on new features by Samsung.

3.6 Discussion

Looking at the data we do see many of the same trends identified by Forrest et al. They found that many corporations seemed to be focused on creating code and were not substantially active within the bug reporting community. We see this same, but with a smaller, but still notable, disparity in the contributor levels. This is most likely due to including the mailing list data and removing the LKML reference email addresses from our data set.

Most telling in this data is that while there are a great number of people involved in some aspect of development, relatively few individuals are active in all three of the areas we studied. Less than 10% of individuals in any of the companies participated in bug reporting, code development, and the community discussion on the mailing list. This corroborates the finding of Forrest et al. that there are very different pictures of participation depending upon the data source used. Any future research into participation in FOSS should take this finding into account.

One interesting point is that there were 15 individuals who changed employers during the timeframe studied. Many of these individuals also changed their activity level in one of the forums after their change in employer. The reasons for this change in activity are unknown, but possibly due to changes in job responsibilities or priorities with the new employer.

When addressing the first research question, we found an interesting split. For the software companies studied, the number of contributors in the git repository was nearly double the number participating in the LKML. This two to one ratio was fairly consistent across all four companies studied as can be seen in figure 2. With hardware companies there was not an apparent pattern, but all had a smaller percentage of employees participating in the mailing list except for Samsung. By adding in the LKML data we are including an additional source of communication within the project. The figures show that while companies may not be involving themselves in the bug repository, almost all have a presence within the mailing list.

Forrest et al. argued that a preference for contributing code without being involved in other communication forums makes code contributions a take it or leave it proposition. They also argued that the data suggested self-focused development practices were occurring in the Linux kernel. Forrest et al provided code commit to bug report ratios greater than 10:1 for many of these companies. Table 6 shows that after accounting for LKML reference addresses, these ratios drop to the range of 3:1 to 6:1 with two exceptions. Not the dramatic difference previously reported, but still a substantially greater number of people involved in developing code than in communicating with the community.

Two of the companies that had greater numbers of bug reporters, Intel and IBM, are interesting. These had the highest ratio of individuals participating in the bug repository compared against the code repository. Of interest here is that nearly 10% of these individuals only participate in the bug repository. Since we chose to include all bug activity, including the reporting of bugs, it could mean that these companies use open source software internally and have a culture that encourages logging bugs in the official forms. Without more evidence we cannot be sure, but it is a good area for future study.

Contrary to the findings of Forrest et al., we found the participation of Fujitsu no longer heavily skewed towards code. Forrest et al found a ratio of over 50:1 for code developers to bug repository participants [14]. After accounting for the obfuscated email addresses, we found a ratio of 5:1; much more in-line with the rest of the corporations. Samsung on the other hand has a high ratio, 34:1, but with only one contributor in the bug repository more information is needed before coming to a conclusion.

Our analysis does lack any knowledge of the internal corporate communication structure and patterns. It is possible that the individuals seen on the LKML are linchpin developers coordinating the activities of large teams behind the scenes at their respective corporations. While we do expect some level of this, we also expect the people developing code to have good reason to communicate via the LKML at some point in the development process. Communicating through another person, a linchpin developer, can add confusion and noise to an otherwise clear discussion. Because of this, we do not expect many companies to hinder their employees' effectiveness by forcing communication through a few key individuals.

It is highly likely that the companies studied here make use of internal bug repositories and may not use the official bug reporting channels. When parsing the commit data we did uncover several messages that acknowledged a bug found by a particular individual, but we could not find a corresponding bug report in the official repository. This suggests that there are other repositories that developers are grabbing bugs from. Fixing any bug in software is beneficial at some level to the community, but doing so outside of the official channels circumvents the work of the management team. This can make it harder to prevent duplication of effort or distribute developer time effectively, both negative outcomes for the project as a whole.

While the number of people involved along with the amount they are contributing gives a high level picture of participation and investment, it does not provide a complete picture, as it does not factor in what developers are contributing. The commit classification data gives us a better understanding of how corporations are focusing their efforts. All the companies are contributing both bug fixes / improvements and new features, but the percentages do have interesting features.

The first thing we see from the data in figure 3 is that Fujitsu is committing more bug fixes or improvements than new features. This is much more in line with the other companies studied. When we couple this data with the bug, commit, and LKML involvement we see that Fujitsu has a much more balanced approach than originally reported.

With one exception, the remainder of the companies have fairly consistent submission ratios. 60%-70% of code contributions are bug fixes or improvements. 10-20% of their contributions were classified as new features. Because of the consistency among these corporations we can consider this a base-line for major corporations contributing to the Linux kernel. Given that the Linux kernel is a mature project, these numbers do not seem unusual.

Samsung on the other hand has a much stronger focus on contributing new features. Over half of all Samsung commits are related to implementing a new feature with only 40% fixing a bug or improving code. While it is certainly possible that Samsung may have less buggy code, we consider that explanation to be unlikely. All of the corporations studied have dedicated software development teams and highly successful products. A difference of the magnitude seen here would be too great to attribute only to better development practices.

While contributing new features can indicate a company’s focus on moving a project forward, coupling this information with the activity in the community gives a much broader picture. Samsung had one of the lowest participation levels in the bug reporting forum and the lowest by far participation in the LKML. Minimal involvement in the community discussion and contributing mostly new features suggests that Samsung has taken a self-focused development philosophy.

One can hardly fault a corporation for acting in its own best interest by following a self-focused development philosophy. This may be good for the corporation, but the benefit to the project may or may not be present. In a sense, if a corporation forges ahead with a self-focused development plan and does not work with the community to coordinate or prioritize, any contributions the make essentially become “take-it or leave it” propositions. The benefit to the company is clear, but any benefit to the project is an afterthought. The fact that Samsung’s changes were accepted into the kernel shows that the project leadership considers them to be beneficial.

3.7 Limitations

With any study of socio-technical artifacts there are limitations. In this section we outline some of the prominent factors. The date range chosen for study was quite large. In this nearly eight year period, the priorities of the companies studied may have shifted. While this may have had an effect on the results, we chose these companies in part because they are substantial and long term contributors to the kernel. We also wished to obtain a sense of longer term trends. In this way, we consider the benefits of the long timeframe to outweigh the potential drawbacks in the data gathered.

Generic email addresses were removed from our data set. This included any addresses from email providers such as gmail.com, hotmail.com, or yandex.ru. While there is substantial anecdotal evidence of sponsored developers using a generic email address, without deeply investigating each contributor, connecting a generic email address to a specific individual would be nearly impossible and fraught with problems. Therefore, removing them from the data set was deemed appropriate.

It is also entirely possible that some members of the community may have participated only in the LKML. In paper we were unable to connect these individuals with a specific corporation and thus they were removed from our data set. While this was not desired, due to the lack of data available we could not provide a mapping. This is an unfortunate, but necessary limitation. Additionally, if an individual is participating only within the confines of the LKML, our arguments concerning alienation still stand as these individuals would not be participating in the greater community.

Some individuals did change employers during the timeframe studied. In these cases we found the corresponding code commits or bug repository contributions that bounded the change in employer and assigned the average of the two dates as the cut-over date. This may not be accurate, but without further information of the employment of these individuals, it is a necessary limitation. Since there were only 15 of these individuals in the data set, we do not expect these to have a significant impact on our results.

The removal of obfuscated email addresses affected some corporations more than others. While all corporations studied except Samsung had some number of obfuscated email addresses in their data set, some saw up to 71% reduction in the number of email addresses attributed to their company. While this may have impacted the totals, it is important to note that all of these addresses refer to a single commit with only one exception—two commits. Additionally, these are not tied to any single individual, they are references to LKML messages to assist in understanding the chain of decision-making. For this reason, we felt our analysis would not be heavily impacted by removing them.

While we did take efforts to classify the commits of each corporation we did not attempt to quantify the value of any commits or bug report contributions. Our data is limited to just a number of commits and our classification. We have also given the same weight to any individual who's name appears on a commit or bug. While this is overly simplistic, it does provide a starting point and without a system for rating these commits and activities it would be difficult to remain objective.

3.8 Conclusions

In following up on the work of Forrest et al. we found similar data, but with the addition of new data sources we were able to resolve with some of the concerns raised in that work [14]. The LKML is often considered the heart of the Linux kernel project and it makes sense that involvement in this forum is important to study when looking at participation as a whole.

We found that companies do have more individuals contributing code to the Linux kernel than are involved in the other sections of the project combined.

This finding agrees with Forrest et al. However, after looking more closely at the data and removing non-personal accounts we found that the disparity is not as large as reported. When activity within the LKML is considered, most corporations do have a presence in community discussions. However, this presence is substantially smaller than the number of individuals contributing to code development. Based on this we take a more cautious stance, but still affirm that corporations are being strategic in where they assign their resources.

In studying the types of code contributions made by corporations we find a baseline of 65% bug fixes or improvements and 25% new features for corporations involved in the Linux kernel. There is evidence of corporations that do not follow this distribution. In the case studied, the types of code contributed coupled with the contribution counts in the other arenas of development suggest this company has adopted a self-focused development philosophy.

Ultimately more study is needed to determine how corporate sponsorship impacts FOSS communities. Understanding this dynamic is important to maintaining openness and trust within the community, especially with non-sponsored contributors.

3.9 Acknowledgements

We would like to thank Amir Azarbahkt, Chad Cooper, and Iftekhar Ahmed for their tireless help in classifying code commits. We would also like to thank the Oregon State University HCI group for their feedback and support of this work.

3.10 References

1.Bakeman, Roger et al. “Detecting Sequential Patterns and Determining Their Reliability with Fallible Observers.” *Psychological Methods* 2.4 (1997): 357–370. *APA PsycNET*. Web.

2.Bergquist, Magnus, and Jan Ljungberg. “The Power of Gifts: Organizing Social Relationships in Open Source Communities.” *Information Systems Journal* 11.4 (2001): 305–320. *Wiley Online Library*. Web. 27 May 2013.

3.Bettenburg, Nicolas, R. Premraj, et al. “Duplicate Bug Reports Considered Harmful #x2026; Really?” *IEEE International Conference on Software Maintenance, 2008. ICSM 2008*. 2008. 337–345. *IEEE Xplore*. Web.

4.Bettenburg, Nicolas, Sascha Just, et al. “What Makes a Good Bug Report?” *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. New York, NY, USA: ACM, 2008. 308–318. *ACM Digital Library*. Web. 25 May 2013. SIGSOFT ’08/FSE-16.

5.Bonaccorsi, Andrea, and Cristina Rossi. “Why Open Source Software Can Succeed.” *Research Policy* 32.7 (2003): 1243–1258. *ScienceDirect*. Web. 2 Nov. 2011.

6.Crowston, K., and B. Scozzi. “Open Source Software Projects as Virtual Organisations: Competency Rallying for Software Development.” *Software, IEE Proceedings -* 149.1 (2002): 3–17. *IEEE Xplore*. Web.

7.Crowston, Kevin, and James Howison. “The Social Structure of Open Source Software Development Teams.” *iSchool Faculty Scholarship* (2003): n. pag.

8.David, Paul A, Andrew Waterman, and Seema Arora. “FLOSS-US: The Free/libre/open Source Software Survey for 2003.” (2003): n. pag. Print.

9.De Souza, Cleidson, Jon Froehlich, and Paul Dourish. “Seeking the Source: Software Source Code as a Social and Technical Artifact.” *Proceedings of the 2005*

International ACM SIGGROUP Conference on Supporting Group Work. New York, NY, USA: ACM, 2005. 197–206. *ACM Digital Library*. Web. 27 May 2013. GROUP '05.

10.Dinh-Trong, T.T., and J.M. Bieman. "The FreeBSD Project: a Replication Case Study of Open Source Development." *IEEE Transactions on Software Engineering* 31.6 (2005): 481–494. *IEEE Xplore*. Web.

11.Ducheneaut, Nicolas. "Socialization in an Open Source Software Community: A Socio-Technical Analysis." *Computer Supported Cooperative Work (CSCW)* 14.4 (2005): 323–368. *link.springer.com*. Web. 25 May 2013.

12.Easterbrook, Steve M et al. "A Survey of Empirical Studies of Conflict." *CSCW: Cooperation or Conflict?* Springer, 1993. 1–68. Print.

13.Elliott, Margaret S., and Walt Scacchi. "Free Software Developers as an Occupational Community: Resolving Conflicts and Fostering Collaboration." *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*. New York, NY, USA: ACM, 2003. 21–30. *ACM Digital Library*. Web. 27 May 2013. GROUP '03.

14.Forrest, Darren et al. "Exploring the Role of Outside Organizations in Free / Open Source Software Projects." *Open Source Systems: Long-Term Sustainability*. Ed. Imed Hammouda et al. Springer Berlin Heidelberg, 2012. 201–215. *link.springer.com*. Web. 27 May 2013. IFIP Advances in Information and Communication Technology 378.

15.German, Daniel M. "An Empirical Study of Fine-grained Software Modifications." *Empirical Software Engineering* 11 (2006): 369–393. *CrossRef*. Web. 24 Oct. 2011.

16.Ghosh, Rishab A. et al. *Free/Libre and Open Source Software: Survey and Study*. The Netherlands: International Institute of Infonomics University of Maastricht, 2002. Print.

17.Gyimothy, T., R. Ferenc, and I. Siket. "Empirical Validation of Object-oriented Metrics on Open Source Software for Fault Prediction." *IEEE Transactions on Software Engineering* 31.10 (2005): 897–910. *IEEE Xplore*. Web.

18.Hars, A., and S. Ou. "Working for Free? - Motivations of Participating in Open Source Projects." *Hawaii International Conference on System Sciences*. Vol. 7. Los Alamitos, CA, USA: IEEE Computer Society, 2001. 7014. *IEEE Computer Society*. Web.

19.Herraiz, Israel et al. "The Processes of Joining in Global Distributed Software Projects." *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner*. New York, NY, USA: ACM, 2006. 27–33. *ACM Digital Library*. Web. 26 Mar. 2013. GSD '06.

20.Hertel, Guido, Sven Niedner, and Stefanie Herrmann. "Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel." *Research Policy* 32.7 (2003): 1159–1177. *ScienceDirect*. Web. 25 May 2013.

21.Jensen, C., and W. Scacchi. "Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study." *29th International Conference on Software Engineering, 2007. ICSE 2007*. IEEE, 2007. 364–374. *IEEE Xplore*. Web.

22.Jensen, Carlos, Scott King, and Victor Kuechler. "Joining Free/Open Source Software Communities: An Analysis of Newbies' First Interactions on Project Mailing Lists." *Proceedings of the 2011 44th Hawaii International Conference on*

System Sciences. Washington, DC, USA: IEEE Computer Society, 2011. 1–10. *ACM Digital Library*. Web. 22 Mar. 2013. HICSS ’11.

23.Jergensen, Corey, Anita Sarma, and Patrick Wagstrom. “The Onion Patch: Migration in Open Source Ecosystems.” *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*. New York, NY, USA: ACM, 2011. 70–80. *ACM Digital Library*. Web. 22 Mar. 2013. ESEC/FSE ’11.

24.Ko, Andrew J., and Parmit K. Chilana. “How Power Users Help and Hinder Open Bug Reporting.” *Proceedings of the 28th International Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2010. 1665–1674. *ACM Digital Library*. Web. 24 Oct. 2011. CHI ’10.

25.Kogut, Bruce, and Anca Metiu. “Open-Source Software Development and Distributed Innovation.” *Oxford Review of Economic Policy* 17.2 (2001): 248 –264. *Highwire 2.0*. Web. 8 Nov. 2011.

26.Lakhani, Karim, and Robert G. Wolf. *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects*. Rochester, NY: Social Science Research Network, 2003. *papers.ssrn.com*. Web. 25 May 2013.

27.Mockus, Audris, Roy T. Fielding, and James Herbsleb. “A Case Study of Open Source Software Development.” ACM Press, 2000. 263–272. *CrossRef*. Web. 6 Nov. 2011.

28.Mockus, Audris, Roy T. Fielding, and James D. Herbsleb. “Two Case Studies of Open Source Software Development: Apache and Mozilla.” *ACM Trans. Softw. Eng. Methodol.* 11.3 (2002): 309–346. *ACM Digital Library*. Web. 22 Mar. 2013.

29.Nguyen Duc, Anh et al. "Impact of Stakeholder Type and Collaboration on Issue Resolution Time in OSS Projects." *Open Source Systems: Grounding Research*. Ed. Scott A. Hissam et al. Vol. 365. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. 1–16. *CrossRef*. Web. 16 Mar. 2012.

30.Raymond, Eric S. *The Cathedral and the Bazaar*. 1st ed. Ed. Tim O'Reilly. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 1999. Print.

31.Robles, Gregorio, and Jesus M. Gonzalez-Barahona. "Contributor Turnover in Libre Software Projects." *Open Source Systems*. Ed. Ernesto Damiani et al. Springer US, 2006. 273–286. *link.springer.com*. Web. 27 May 2013. IFIP International Federation for Information Processing 203.

32.Sandusky, Robert J., and Les Gasser. "Negotiation and the Coordination of Information and Activity in Distributed Software Problem Management." *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*. New York, NY, USA: ACM, 2005. 187–196. *ACM Digital Library*. Web. 24 Oct. 2011. GROUP '05.

33.Scacchi, W. "Free and Open Source Development Practices in the Game Community." *IEEE Software* 21.1 (2004): 59– 66. *IEEE Xplore*. Web.

34.Scacchi, W. "Understanding the Requirements for Developing Open Source Software Systems." *Software, IEE Proceedings -* 149.1 (2002): 24–39. *IEEE Xplore*. Web.

35.Schilling, A., S. Laumer, and T. Weitzel. "Who Will Remain? An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects." *2012 45th Hawaii International Conference on System Science (HICSS)*. 2012. 3446–3455. *IEEE Xplore*. Web.

36.Stewart, Katherine J., and Sanjay Gosain. "The Impact of Ideology on Effectiveness in Open Source Software Development Teams." *MIS Quarterly* 30.2 (2006): 291–314. *JSTOR*. Web. 25 May 2013.

37.Von Krogh, Georg, Sebastian Spaeth, and Karim R Lakhani. "Community, Joining, and Specialization in Open Source Software Innovation: a Case Study." *Research Policy* 32.7 (2003): 1217–1241. *ScienceDirect*. Web. 26 Oct. 2011.

38.Ye, Yunwen, and K. Kishida. "Toward an Understanding of the Motivation of Open Source Software Developers." *25th International Conference on Software Engineering, 2003. Proceedings*. 2003. 419–429. *IEEE Xplore*. Web.

Chapter 5. General Conclusion

We found that for these large projects, corporate developers dominate in terms of code contributions. This has important implications for project governance and our understanding of FOSS demographics. This finding does not appear as dramatic with the addition of mailing list participation data, but is still well substantiated.

The data suggests there exist at least three distinct communities within projects. While these communities may interact with each other through other means (e.g. irc, direct email, or wiki pages), there is a community of coders, a community of bug reporters, and a community of mailing list participants. While this is not unexpected, it is unexpected to see that very few contributors participate in all three forums and the proportion that participate in only one forum is quite large. This disconnect can in the long-term lead to alienation and declining participation of non-technical contributors.

Our second investigation method, which included the addition of mailing list participation data and classification of code commits, provides a more nuanced and likely more accurate picture of participation by corporations. In studying the types of code contributions made by corporations we find a baseline of 65% bug fixes or improvements and 25% new features for corporations involved in the Linux kernel. There is evidence of corporations that do not follow this distribution. In the case studied, the types of code contributed coupled with the contribution counts in the other arenas of development suggest at least one company has adopted a self-focused development philosophy.

We also found that many projects do not currently track this kind of data, or at least they do not make it publicly available. While there may be privacy concerns with posting email addresses or calling out individual developers or companies, this has to be balanced against users and other contributors’ need to know. Without this information, FOSS users and possible contributors lack the necessary information to understand whether a project is well governed and healthy.