Exploring the Role of Outside Organizations in Free / Open Source Software Projects

Darren Forrest, Carlos Jensen, Nitin Mohan, Jennifer Davidson

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

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

School of EECS, Oregon State University, Corvallis OR 97331, USA

{forresda, jenseca, mohanni, davidsje}@eecs.orst.edu

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, NGO's 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.

Keywords: Governance, Contributor affiliation, Participation metrics, Community sustainability

1 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 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].

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 (http://pypi.python.org/pypi/publicsuffix) to consolidate domains. The crowd-sourced public suffix effort by Mozilla helped us effectively collapse 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.

4 Results and Discussion

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].

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.

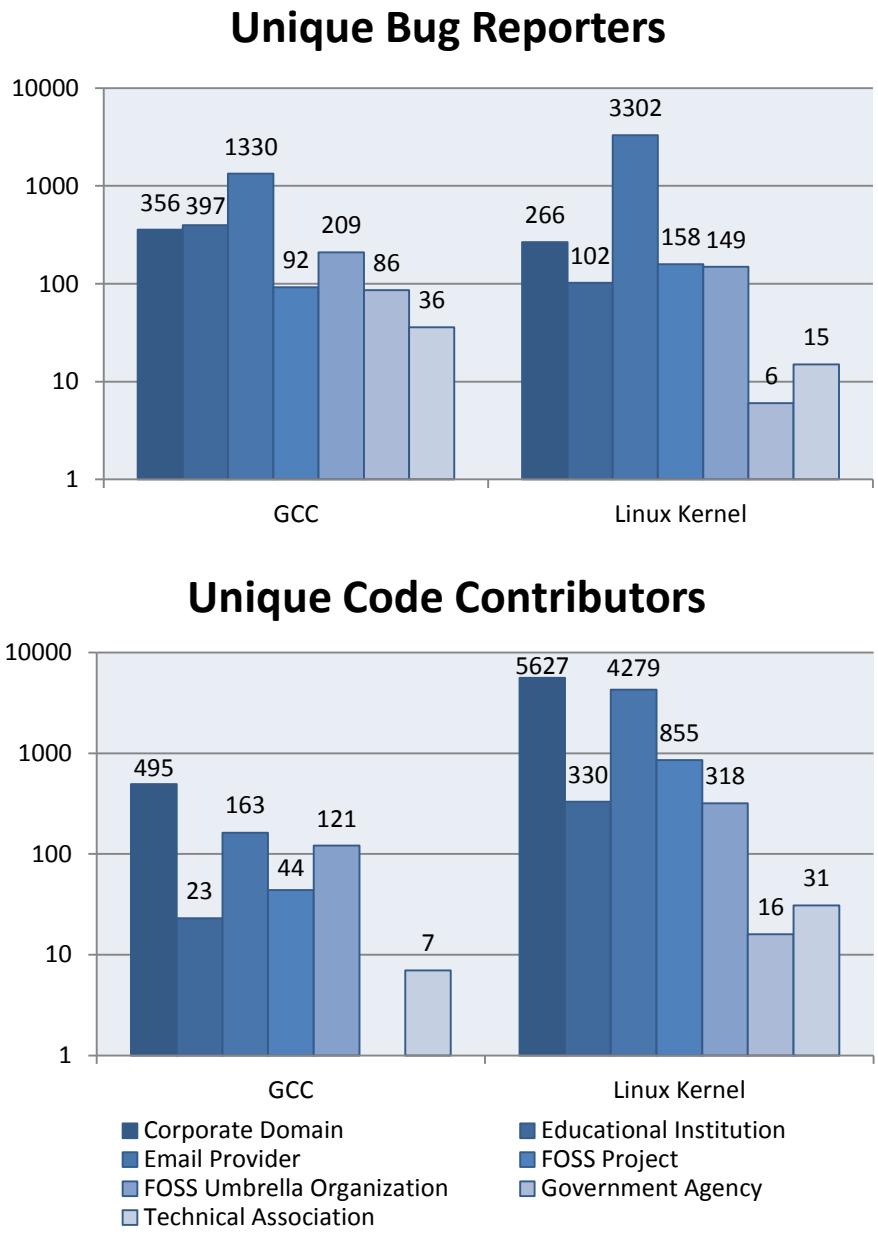

Fig. 1. Categories of participation for GCC and Linux Kernel. Logarithmic scale

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 develop-

ment 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%.

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

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

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 ac-

tivity, as some organizations can have few people contributing a lot, or a lot of people contributing very little. 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.

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

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

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.

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.

Table 3. GCC contributions ordered by coder/bug reporter ratio

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

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.

Table 4. Linux Kernel code commits to bug reporting ratio

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

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.

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.

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.

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.

Acknowledgements

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.

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. Standford Institute for Economic and Policy Research, Standford, 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