Submission: 15601

THE IMPACT OF COMMERCIAL ORGANIZATIONS ON VOLUNTEER PARTICIPATION IN AN ONLINE COMMUNITY

Keywords: Computer-mediated communication and collaboration, Virtual teams, Software development methodologies, Longitudinal research, Network analysis

ABSTRACT

Open Source software is widely seen as a key component in managing complexity and risk in commercial software development. However, the ramifications of the introduction of commercial developers into an existing volunteer managed project are not clear and this represents a risk for firms participating in Open Source communities. Many of the oldest and most prominent Open Source projects consisted of loosely coupled volunteers contributing ideas and code to create software with great utility for the developers with little regard to the possibility of personal financial gain. When a commercial firm participates in volunteer founded and managed projects their developers may take on various roles in the project, but volunteers often remain a key component. To address whether commercial participation increased or decreased volunteer participation in Open Source communities we conducted a combination qualitative/quantitative study with interviews of Open Source developers and found that in most cases participation by commercial developers attracts volunteers but that this effect is strongly moderated by the firm's overall strategy in relation to the project and community. We found that firms offering support services or contributing across the entire community typically increase the number of volunteer participants, while firms focused on a niche component within the community, typically have statistically insignificant effects. These findings suggest that commercial firms that seek to participate in an Open Source project should choose to participate and contribute across an entire community rather than a single niche component.

INTRODUCTION

Modern software development projects are complicated, prone to failure, and loaded with uncertainty. Frequent time and budget overruns doom projects to obsolescence before they even get a chance to launch (Brooks, 1995). Because of the rather liberal licensing terms of Open Source that require no payment for use, distribution, and modification of the software (Rosen, 2004) and partially in effort to reduce risk, budget, and time to delivery, commercial firms frequently adopt portions or the entirety of Open Source projects as components of their product (Bonaccorsi, Giannangeli, & Rossi, 2006; Goldman & Gabriel, 2005; Krishnamurthy, 2005). For example, the Android operating system from Google and Symbian operating system from Nokia are both Open Source platforms that cell phone manufacturers utilize to create the latest generation of smart phones (Lawton, 2008). While Android and Symbian directly face the end user and both have their genesis from a commercial effort that was later released as Open Source, numerous devices utilize portions of Open Source technologies as core components that are hidden behind the scenes, such as TiVo, Amazon's Kindle, and the Nokia n900 series of cell phones (Bartholomew, 2008; Barton, 2003; Maxwell, 2006).

Historically, these components often arise from volunteer communities that began without direct commercial goals. Rather the communities began from the shared interest of a community of software developers collaborating over the internet (Moody, 2001; Raymond, 1999). As these Open Source projects have evolved and gained in overall utility many have attracted a variety of developers from commercial firms who are paid to work full time on the project. In the Firefox web browser, Linux operating system, and OpenOffice.org suite of office programs, volunteer and paid developers from numerous firms collaborate to plan and develop key features of the software (Fielding, 2005). These firms and their associated developers may have different goals from those of volunteers and in some cases may behave in ways that are inconsistent with the standards and norms of the volunteer Open Source

community. Their presence in Open Source projects could either foster or disrupt the original volunteer communities. Aside from getting bad reputation in the broader Open Source community, disrupting an Open Source community would almost certainly result in increased work and costs for the firm as it shoulders some of the burden that was previously managed by the volunteers. While previous research has addressed the motivations and actions of individual commercial developers in an Open Source software environment (Bitzer, Schrettl, & Schroder, 2007; Lakhani & Wolf, 2005; Shah, 2006), there has not been any analysis of the overall community impact of commercial participation (Oreg & Nov, 2008; Roberts, Hann, & Slaughter, 2006). The problem is further clouded by conflicting theories that suggest that group identity may increase participation, while the increased heterogeneity from commercial developers would decrease participation by volunteers. The primary goal of this paper is to address such conflicting theories and understand how volunteer participation in Open Source projects changes in the presence of developers working for commercial firms and further to identify attributes of commercial participation that may have an impact on volunteer participation. In the following sections we describe four distinct routes through which commercial firms could influence volunteer participation in Open Source projects and communities.

Commercial Participation and Positive Project Momentum

When a commercial firm first enters a volunteer Open Source project existing project members may see their presence as a benefit to the community. Commercial firms often provide wider distribution by adding the project to their existing offerings, garner media attention for the project by issuing press releases about the software, and present information about the project and community at trade shows. For many developers who identify themselves with the project this may be seen as a sign of success, increasing their willingness to continue to contribute to the project (Ashforth & Mael, 1989). Furthermore, when participating in the community many companies encourage their employees to become active in the community through the use of community run mailing lists and websites such as

wikis, bug trackers, and weblogs. These community tools, while providing a method for commercial developers to become more aware of the project and minimize their impact, were designed to provide a mechanism for developers to discuss, plan, and share what they're working on. Individuals outside of the community can easily see this activity about the current happenings in the community, leading to a perception of an active and successful community and an incentive for new developers to become involved (Hars & Ou, 2002). An individual who is only marginally involved with the community and looking for ways to get involved may see the participation of commercial firms in these tools as validation of the project and seek to participate and contribute to such a project because of the possibility for future rewards, such as increased technical know-how or the possibility of career advancement (Roberts et al., 2006).

If commercial developer participation validates the importance of the project, and increases the overall project momentum, then an influx of commercial firms and paid developers should attract volunteer developers and increase their participation in the community. The number of changes that full time commercial developers can make, and their high level of skill may speed up the development process, increasing the utility of the project to community members and contribute to volunteers' identification and attachment to a successful project (Hertel, Niedner, & Hermann, 2003; Polletta & Jasper, 2001). Such attachment with projects, communities, and movements is an important factor in volunteers remaining active in a community and overall community success both in conventional volunteer organizations and Open Source communities.

Hypothesis 1: The participation of commercial developers on an Open Source project will increase volunteer participation in the project.

Negative Impacts of Heterogeneity

However, just as participation by commercial firms can provide resources to a community, they also introduce heterogeneity into the pool of developers. Whereas initially all of the volunteer developers may have been able to rally around the primary focus of the community, developers employed by commercial firms may be working just for a pay check, with little concern for overall community health and well-being. In the long run, such heterogeneity in workgroups decreases overall productivity and increase tension within teams (Pelled, Eisenhardt, & Xin, 1999; Williams & O'Reilly, 1998). Furthermore, these developers may have different work habits, communication patterns, and skill sets, which introduce heterogeneity into the work process with possible negative effects (Jackson et al., 1991; Star & Griesemer, 1989). Open Source communities have additional issues of heterogeneity which result in decreased performance, such as personal ideology for community participation (Stewart & Gosain, 2006).

Hypothesis 2: The participation of commercial developers on an Open Source project will decrease volunteer participation in the project.

Business Models and Community Norms

In the late 1990's when Open Source was first attracting interest from commercial firms, most had similar business models. These firms followed the model of Linux distributors, such as Red Hat, that took the output of the community as a whole, packaged it with documentation and additional software to make it easier to use, and then sold the complete collection of software with enhanced support (Tiemann, 1999). These companies tied their financial success to the success of the Open Source community as a whole. More recently as the market has matured, additional business models have arisen that allow firms to isolate and derive revenue from a single component from a larger community or start their own communities around small niche products (Krishnamurthy, 2005).

We identify two broad classes of commercial firms based on their business model in relation to the community: community focused firms that package the entire output of a community, such as Linux distributors, and product focused firms that utilize only a portion of the output from the community for their products. Product focused firms typically enhance particular components from a community, such as a component library, or focus on a particular application in their business model. As many of the volunteers identify with the entire community of projects, rather than a small component in a project, it is likely that community focused firms have less of tendency to increase heterogeneity in the community that product focused firms and, as a result, will be viewed slightly better than product focused firms, leading to an increased power in attracting new volunteer developers.

Hypothesis 3: Developers from community focused firms will be associated with a greater increase in the number of volunteer developers than the presence of developers from product focused firms.

Cognitive Complexity at the Module Level

At the heart of Open Source projects is source code – files written in various programming languages that embody the primary functionality of the project. Techniques for managing Open Source have continually evolved, with more advanced programming languages and improved software engineering practices that often mirror, or in some cases are the predecessors, for widely accepted commercial practices (Neus & Scherf, 2005). The code for complex projects may consist of hundreds or thousands different files, each performing a specific task. To assist in developer logistics and comprehension, within large projects code and responsibilities are typically divided into smaller components, called modules (Parnas, 1972). For example, a simple email client may have three modules: receiving mail, sending mail, and graphical user interface. All work within a module must typically be carefully coordinated, since all parts of the module tend to be closely coupled. Work in different modules tends to be much more loosely-coupled, and typically requires much less coordination (Baldwin & Clark, 2000). Each module may have a set of developers who are responsible for maintaining the module and overseeing development.

Organizationally, modules often replicate the structure of the larger project – complete with their own mailing lists, bug tracking, and social norms (German, 2005).

Because of the distributed nature of most Open Source software development, projects have adopted strong norms of open communication and decision making. For example, the Apache project requires that all decisions reach consensus on publicly accessible mailing lists (Fielding, 1999). However, without explicit controls collocated developers employed by a commercial firm who work closely together will have decreased incentive to post to the project mailing lists and maintain the transparent decision and documentation process (Campagnolo, Tacla, Paraiso, Sato, & Ramos, 2009). Such a process increases the cognitive complexity of code and prevents volunteers from fully understanding the logic of the new code. The loss of open discussion allows collocated developers to create code that is less modular making future changes more difficult further decreasing participation (MacCormack, Rusnak, & Baldwin, 2006). Because commercial developers work full time, they change project code much faster than volunteer developers. A survey of volunteer Open Source developers found volunteers average 14 hours a week on Open Source projects only a third of a standard 40 hour work week for commercial developers (Lakhani & Wolf, 2005). These issues pose a real danger that as developers from commercial firms modify code within a module of a project, it will become increasingly difficult for volunteer developers to comprehend the set of the changes, forcing the volunteers previously working on the module to migrate to alternate modules within the project or leave the project completely.

Hypothesis 4: The participation of commercial firms in modules of an Open Source project will reduce volunteer participation in those modules.

Our goal in this paper is to test each of these hypotheses and identify the net effect of commercial developers on volunteer developer participation and to understand the mechanism through which this occurs.

RESEARCH METHOD

Open Source software projects usually have rich historical archives of communication thanks largely to the fact that commonly accepted best practices use project support tools to automatically record every communication, debate, and decision. For most projects the tool of choice for this is simple email lists that are automatically archived, other projects utilize archived real time chat, or web based forums for communication. The process of ensuring that all communication is open and archived is common to most Open Source projects (Fogel, 2005). This archived data provides a rich resource for researchers; however, it may not provide the complete context of decisions and communications to researchers or participants without in-depth knowledge of the project. To ensure that we were correctly interpreting the data, we performed a complimentary qualitative analysis of developers and projects. This allows us to understand the nuances of how communities and commercial firms interact and is particularly helpful in the cases where commercial firms make decisions regarding project participation outside the framework of the project. Our research utilizes a multi-method approach to understanding these questions around commercial participation in Open Source software communities

We conducted two studies focusing on a single large Open Source community. The first study was a qualitative study to identify the views of developers toward commercial participation and to provide additional background context about the community. The second study analyzed quantitative data obtained from the community in order to evaluate hypotheses regarding commercial participation in Open Source suggested both by previous research and the results of the qualitative study.

Community Background

Our research focuses on the GNOME project, a large and successful Open Source desktop environment started by volunteer developers in 1997 as a response to the lack of a completely free and Open Source desktop environment for Linux and other free computer operating systems. By many metrics, this is a

highly successful project: more than 10 years of history, stable releases every six months, and a continually growing user base (de Icaza, 2002). It is the desktop environment for computers from Sun Microsystems, software from the project is in use in a myriad of devices like the One Laptop Per Child and Nokia n800 series of Internet tablets, numerous startup firms have created solid businesses around the project, and is available direct from many computer manufacturers including Dell computers as part of their option to provide Linux on new computers (The GNOME Foundation, 2010). In our period of analysis, which goes from the origins of the project in 1997 to late 2006, there were over 1200 individuals who had "commit" status – the ability to directly modify the project source code without needing to go through an intermediary – and almost 1000 different components in the shared source code repository. The community coordinates most of their activity through Internet enabled tools such as a shared bug tracker, mailing lists, and real time chats. The community, although originally comprised only of volunteers, has adopted modern software engineering practices (German, 2004) and faces many of the coordination and collaboration issues found in most distributed teams (Koch & Schneider, 2002).

One of the key elements of the GNOME project is that it is comprised of many smaller projects of varying size, complexity, and maturity. For our purposes, when we refer to the "GNOME project" we mean this larger community, and a "project" is one of these smaller projects in the community. The community acts in a federated manner, providing each project with the opportunity to control their own outcomes and chart their own roadmap subject to some broader constraints and goals developed by the community (German, 2005). When a project has reached sufficient maturity, the developers may apply to have that project included as part of the main community software distribution, greatly increasing the probability that the project will be included as a default component with new installations of Linux and granting the project a large user base. Most projects in the community have their own mailing lists and bug trackers and are generally managed by individuals working on those projects. Most community

participants are active on multiple projects, but because there are hundreds of projects within the community, there are no developers who are actively contribute to, or able to monitor all the projects.

The community has a track record of commercial investment – during the dot-com boom of the late 1990's several firms were created to customize the project, develop components, and provide support for users of the software. However when the first dot-com bubble burst in 2000-2002, many of these firms went bankrupt or left the market, leaving critical components largely unmaintained. The community slowly built up commercial support again and now has significant corporate investment from firms that distribute software as a component of the Linux operating system, and from other firms that utilize the software as a base toolkit that can be used for the design and manufacture of embedded devices such as PDAs and mobile phones.

STUDY 1: QUALITATIVE DEVELOPER INTERVIEWS

To obtain and better understand the context of the community and the relation between commercial and volunteer developers we began our overall study by first conducting a small qualitative study of developers from the GNOME community. The first author attended one of the two major annual face to face meetings for both volunteer and commercial community participants. These events are generally considered to be one of the highlights of the year for the project and take place shortly after the major releases of the software, approximately every six months. To encourage participation by volunteers in the conferences, the GNOME Foundation, which oversees many of the legal and marketing aspects of the community, provides travel stipends to volunteers in the community to attend the conferences. While this helps volunteers attend the conference, because of issues with getting time away from work or school and the limited number of stipends, volunteers are typically underrepresented at the conferences.

Before attending the conference we obtained a list of attendees and contacted a variety of individuals in the community. We began by identifying the ten attendees with the most commits to the community source code repository, next we identified those individuals who had official named roles in the community beyond their project (for example the community "release manager" who worked with all projects to ensure that the overall community released a coherent set of projects in bulk once every six months), finally we used random sampling to identify ten more individuals not in the previous set. In total 27 of the conference attendees were contacted, twenty replied indicating they would be willing to be interviewed. During the course of the conference, eighteen of the twenty identified individuals were interviewed over the course of three days (one individual was unable to attend the conference, and another left after the first day before an interview could take place) . Interviews were semi-structured, lasted twenty to forty minutes, and were conducted during breaks in the schedule. Each interviewee was asked for the relevant professional background, how they got involved with the community, where they currently participate in the community, how they relate to commercial developers in the community, and if they believed our firm classifications were accurate.

General descriptive statistics about the interviewees can be found in Table 1 below. Fourteen of the eighteen individuals were committers who could directly commit changes to the project source code repository– two of the newer developers, one commercial and one volunteer, still needed to contribute through intermediaries and neither of the individuals who self-described their role as community support could make changes directly to the source code. As is typical in Open Source all the interviewees were male (Ghosh, Glott, Krieger, & Robles, 2002).

Insert Table 1 about here

The nine volunteer participants had varied backgrounds. Three of the volunteers identified themselves as students who primarily participated during their free time. The other six volunteers indicated their use

and participation in the project was related to their roles at work. All six of these non-student individuals contributed to projects in the community while they were "on the clock", even though participation in the community was not an official part of their jobs. These participants believed their participation was relevant to their jobs and participation improved their performance at work. The participants came to the project through a variety of routes. Most first became interested in the community because of their general interest in Linux and technology, but their reasons for changing from a passive community member who only uses the software to an active, contributing, member varied. Three of nine volunteer developers indicated that another individual working in the community had played a very large role in bringing them in to work on the community. Two of the developers indicated they started submitting changes to a project in the community and were later offered the chance to become maintainers of the project. The remaining four volunteer developers could not identify a specific reason they became more active in the community. Five of the commercial developers were active as volunteers in the community before they were hired. The remaining four commercial developers were hired by the firm for other projects and later shifted to projects in the community.

Views of Commercial Participation

Both commercial and volunteer developers thought commercial developers provided manpower and the focus necessary to accomplish tasks that volunteer developers lacked the skill or motivation to accomplish. Volunteers often indicated that they enjoyed working on projects with commercial developers because the projects tended to be widely used and those volunteers saw adoption of their project as a key indicator of success. Several commercial developers believed their firms provided a marketing force for the community, increasing the appeal and bringing in more individuals to work on and participate in the community. None of the developers, volunteer or commercial, ever mentioned intentionally treating another individual differently because they worked for a different firm or were a volunteer, although a commercial developer did indicate that he believed code written by volunteers

wasn't always as useful as code from his firm. This set of statements from community members lends general support for hypothesis 1 and against hypothesis 2.

"If it weren't for [commercial developer name], I wouldn't be involved in the community. He saw my postings on the mailing list and encouraged me to get more involved. About a month later he asked if I would like to maintain the project."

– Volunteer developer speaking about how he became involved

However, two of the commercial developers who began working in the community as volunteers expressed a small amount of frustration in aligning the goals of their firm and the community – possibly alienating volunteers in the community, but generally thought their firms had found ways to succeed. In one case the firm adopted a dual process model for participation in Open Source, where developers had individual responsibility for ensuring their participation was congruent with the values and norms of both their firm and the project. This model caused problems for the firm with documenting work and assigning work – especially in the case of volunteers working on a component that the firm and volunteers had differing design goals. The other developer indicated that when his firm first became involved with Open Source, they intentionally restructured the norms of the development team to align with the community and not the firm. Although this caused contention within the firm, it was thought to be best for the community. This indicates that, at least to some degree, developers viewed it as possible that commercial participation could drive away volunteer developers, as indicated in hypothesis 2.

"I certainly would not want to see commercial participation go away. But I think there are things that some companies should be more careful of when working in the community. At [firm name], we've been very careful how we work with the community."

– Developer at community focused firm

The need to be careful when choosing how to participate was echoed by a commercial developer who had been active in the community for more than five years and had worked with multiple firms. He was currently employed by a product focused firm, was critical of his firm's participation, believing that his current firm had little respect for the community norms. Rather he believed his firm was involved only the sake of exploiting the community for their own products, and had little interest in the health and values of the overall community. This view was in sharp contrast to his previous experience at a community focused firm that he described as fostering involvement within the community. This developer left the product focused firm and the entire community shortly after the conference and his departure stirred up debate within the community about how firms should interact with each other and volunteers. This indicates there was some support for hypothesis 3 amongst the developers.

Three of the volunteer developers believed there were times when the increased skill and time dedication from commercial developers was beneficial, particularly in highly technical areas such as system performance and low-level libraries that volunteers often could not develop. The volunteers did not indicate that they felt that this had ever caused them to abandon work on a module because they indicated they lacked the skills to work on such complex issues themselves. This set of statements goes against hypothesis 4.

Classification of Firms

Prior to the interviews we classified each of the nine largest firms, as measured by number of source code commits, according to their business model relative to the community: firms that focused on a single project or small subset of projects in the community were called product focused, while those that supported the entire community were community focused. This classification was done through extensive observation and study of the business models of firms in the community and supported by a principal

component analysis on attributes obtained by mining the archival data from the community. A brief description of each of these firms can be seen in Table 2. Each of the interviewees was provided a description of our classification scheme and asked to classify each of the nine firms as either a product or community focused firm. Out of the 162 classification tasks across interviewees, only two were not in agreement with our classification (Fleiss' κ =0.953). Incidentally, the two points of disagreement were both employees of a product focused firm who believed their firm was community focused.

Insert Table 2 about here

However, when asked about their emotional perception` of specific firms the views of interviewees varied. Attitudes were almost universally favorable of community focused firms. In contrast, product focused firms had mixed perceptions from developers and, in the words of one volunteer, were guilty of "not caring about volunteers." Another volunteer who maintained a project within the community that had contributions from about ten developers was extremely skeptical of participation by a major firm, despite being good friends with many of their developers and contributing to other projects maintained and stewarded by the firm. He expressed concern about the method of participation by the firm and the fact that they didn't require everyone to go through the same community socialization process before gaining committer status. This led him to be wary of contributions to his component from the commercial firm. Later in the interview process, when five of the other volunteer developers were asked specifically about this firm, they all echoed similar concerns about the firm's participation.

"I don't think the commercial firms have the same interests as volunteers. If they submitted code to my project, I'd accept it, but if they started to submit lots of code, I'd start to look a lot more at where the project was going."

– Volunteer developer and project maintainer

These interviews paint a mixed picture with regards to commercial development. While most developers indicated that they appreciated commercial development in the community, a substantial portion of the developers were skeptical about the behavior of these commercial firms. The views expressed toward commercial firms by the developers indicate that there may be a relationship between business model and perceived attractiveness of commercial firms in the community.

STUDY 2: QUANTITATIVE ANALYSIS

Theory surrounding the issue of commercial participation in Open Source communities and the interviews conducted in the first study provide a foundation for our second study, a longitudinal analysis of historical data obtained from the community. We begin by further validating the classifications by business model proposed to, and validated by, the interviewees through an analysis of three kinds of behavior of commercial and volunteer developers. Within the community there were three different ways that participation and actions were automatically archived: 1) project mailing lists that were open to everyone, both newcomer and experienced participants, that addressed technical and non-technical issues, 2) a publicly accessible project bug tracker that was open to anyone, but required a moderate amount of technical skill and knowledge to interact and contribute to, and 3) project source code which was available to everyone, but accepted direct contributions only from experienced and knowledgeable community members called "committers" – other community members who wished to contribute to the project source code needed to post their contributions to the project mailing list or bug tracker and then have a committer make the modifications to the source code. These three different data streams are common to nearly every Open Source project, were found in each of the projects in our study, and represent the common methods by which anyone from a high school student who has just discovered the project to a seasoned commercial software developer contribute to and improve the software. The data are then used to develop profiles of developers and projects and test for the effects of commercial participation on volunteer participation at the project and module levels.

Data Collection and Analysis

Most Open Source projects follow a set of norms that are generally referred to as the Open Source process. A key component of this process is the collection and archival of nearly all communication data as a form of organizational memory and as a tool for developers and users to later reference. In most communities public mailing lists are archived where they can be easily indexed and searched, bug tracking systems provide a complete audit history of every change made to each bug report, and a version control system manages and records all changes made to the software. When using a version control system, each developer downloads a complete copy of the code for the project, makes and tests their modifications, and then sends information about the files that were modified back to a main server in a single action called a commit. Each change is tracked in the system, allowing developers to revert to a previous point in the development process or "roll back" changes that may have been detrimental to overall development while providing a method of providence for all code modifications (Fogel, 2005).

Working with the system administrators in the community, archival copies of mailing lists, bug databases, and the community's version control system were obtained. During the initial period of study, the community utilized concurrent version system, CVS, as their version control system and later switched to Subversion. It was set up in such a way that any developer with a CVS or Subversion account could commit directly to the repository for any project. To control the community and protect the project source code, access was only granted to developers after a request was made by another developer to create the account, limiting the number of individuals who could contribute to those who have actively shown interest and potential to improve the project. Bugs and requests for enhancements were managed and tracked using the Bugzilla software package, allowing anyone with an account, available instantaneously through a web form, to submit and comment on issues related to a project. The mailing

lists were managed using the Mailman software, and most lists allowed anyone with an email address to subscribe and participate in discussions.

Each of the tools utilized different account and identity management solutions. All accounts belonging to each developer were manually unified and linked together within the data set. Frequently there were obvious cases where accounts could be linked, such as a consistent username or email address. Variations on names and email addresses were also searched out, identifying shorter versions of names, consistent email addresses across domains (e.g. developer321@hotmail.com and developer321@gmail.com), and small changes within domains (e.g. john.smith@company.com and j.smith@company.com). Where accounts were difficult to match up advice was sought out from members of the community who worked on the same set of projects as the problematic account. Finally, at the end of the process a random collection of the accounts were spot checked by members of the community for accuracy. This allowed us to simply and directly obtain all contributions for developers across different projects and mediums. Information about developers was augmented with employment information gathered from examining developer email addresses, signatures at the end of messages, blog postings, project web pages, and interviews with community developers. In many cases it was possible to calculate exact dates that a developer began working for a firm thanks to blog posts, changes in email, or other public notification. Developers in the community also helped with this classification task when the commercial affiliation of a developer could not be easily found. This provided the necessary information to classify a developer's participation as volunteer, product focused commercial, or community focused commercial allowing the analysis of firm level behaviors in the community.

Product Focused vs. Community Focused Developers

The interviewees supported our idea that there were two different types of firms in the community – community and product focused. Initially, based on some of the comments of the interviewees, we

believed that community focused developers may have more experience and thus be seen as experts in the community. As a result of this increased expertise, community focused developers may be more likely to have a positive effect on volunteer participation because their beliefs and actions had more time to line up with those of volunteers in the community, or because the volunteers may see these developers with increased experience as elder statesmen and seek to work with them. This seemed reasonable, as the first firms to invest in the community were community focused firms. However, there was no statistical difference between the tenure in the community for product focused (mean of 5.24 years) and community focused (5.51 years) developers. Both had more experience in the community than did volunteers (3.93 years).

Several interviewees also believed that there were observable behaviors that differed between community and product focused developers. Based on interview responses and personal experiences in the community we identified and analyzed a set of behaviors that may be seen as pro-social and community building. These behaviors have the primary characteristics of showing an interest in the community beyond the narrow focus of products the developer is paid to work on or are behaviors which have a high probability of interacting with individuals in the community who are not already developers. As developers were active in the community for widely varying amounts of time, we normalized each level activity by the number of years the developer was active in the community.

The first way that individuals outside the community are likely to interact with commercial developers is through project mailing lists. Individuals that start many new discussions and reply to a variety of messages are likely to interact with a variety of volunteers and address issues raised by the community. Beyond being highly active, the number of mailing lists a developer posts to also increases the sense that the developer is building community, especially if the developer is active on mailing lists of projects for which they have not written code.

We examined the mailing lists from the community and for each developer counted the number of messages posted, new discussion threads started, projects mailing lists they were active on, and the number of projects they posted messages to for which they had never contributed code. Each of these values was normalized by the number of years the developer had been in the community, as measured by the duration from their first observable contribution in any project to their last observable contribution (or the end of the data set if still active). In addition we examined the messages posted by the developers looking for pro-social behaviors such as providing references to another developer's email address and post links to websites, indicating that the developer was attempting to redirect and guide other individuals. We then performed an ANOVA to compare among the classes of developers: volunteer, product focused, and community focused. The results as shown in Table 3 indicate that there is a significant difference in participation patterns between the three classes of developers. In particular, commercial developers were found to be much more active on mailing lists. However, when tests were performed analyzing just the difference between product and community focused commercial developers; a difference was found only in the number of messages posted to project mailing lists.

Insert Table 3 about here

Further content analysis of all email messages sent to public mailing lists identified elements that support information seeking behavior in the community – posting email addresses of contacts and providing pointers to web pages. The results were then aggregated by whether the author of the message was employed by a community or product focused firm. This analysis found that community focused developers posted references to email addresses 86% more frequently than product focused developers, and web addresses 161% more often. Both of these behaviors are pro-social community building behaviors that help new community members become acquainted with the project and participants.

Mid-level technical interactions on the Bugzilla bug tracking system may have similar affects in building community as posting to message to a mailing list. In particular, users are encouraged to post any bugs encountered to Bugzilla. These bugs are periodically triaged by a group of community members who then assign the bugs to project developers. At the simplest level, each bug is given a small message form that allows developers to post messages that are sent back to the original submitter and any other individual with interest in the bug. Often times, developers post messages indicating that a bug has been verified as present, asking for more information, or provide a workaround for the user experiencing the bug. Individuals also may submit patches to bugs that address the bug, a behavior that could be seen by a volunteer as taking a significant interest in their issue. When working with Bugzilla, developers can mark bugs as "fixed", indicating that the patch has been submitted and accepted. Finally, we can count the number of distinct projects a developer was active on within the Bugzilla system to get an idea of overall activity and also cross-reference this against activity in the source code repository to see where developers contribute to bug management but do not write code. We take this to be an indication that a developer is taking concrete action showing a broader concern for the project, beyond the local areas that are the focus of the developer's interest.

We collected each of the previously described metrics for every developer and used the same method as we did for the mailing lists to normalize for the length of time the developer was active in the community. The metrics were aggregated by class of developer, and summarized in
Table 4. Surprisingly, while commercial developers had a greater frequency of activity, as measured by the number of comments, patches, and bugs fixed, they had the same relative amount of breadth in the system as volunteer developers. Contrary to our initial belief, when accounting for tenure in the project, product focused developers were active on project bug trackers for significantly more projects and projects for which they had written no code than community focused developers (as measured by the Extra Projects variable). However, as a whole, the ANOVA for these values were not significant.

Insert

Table 4 about here

Certain actions by commercial developers in the CVS code repository may also be construed as pro-social community oriented behavior. Working on a variety of projects within the community shows that the developer has a greater interest in the overall health and well-being of the community. Analysis of the logs indicates that developers employed by community focused firms are active on significantly more projects, as shown in Table 5. The increased participation across a wider number of projects confirms the responses by many of the volunteer interviewees who believed the product focused firms worked only in narrow niches within the community and that community focused firms spread their effort across multiple projects.

Insert Table 5 about here

This analysis shows that there are sometimes dramatic differences in the patterns of participation between volunteer, product focused, and community focused developers. While, in general, all commercial developers are more active in the community than volunteer developers, community focused developers are much more active and visible on community mailing lists and within the project source code. However, product focused developers tend to be more active in the moderately technical realm of bug tracking, indicating that such work may be critical to the success of their business model.

Quantifying the Impact of Commercial Developers on Volunteer Participation

The community makes it very easy for developers to start a new project, leading to a variety of projects that contain only small amounts of code, or represent the efforts of only a single developer working on a very specific tool. To select projects that had a substantial community around them and enough data to achieve statistical significance, we filtered the data selecting only projects with more than 20 developers, more than 100 bugs filed in the Bugzilla bug tracking system, at least one community hosted mailing list

associated with the project, and more than a year of overlap between the source code history, mailing list archives, and bug tracker data. These requirements yielded 14 projects from the community. To avoid problems related to temporary changes in behavior, for example, the "freezing" of the project code right before a release when only critical fixes are allowed into the project source code repository, the data was broken up into 8 week periods. At each period the number and identity of volunteer and commercial developers committing code during the period and the number of commits to the project during the period were recorded. Summary statistics and correlations can be seen in Table 6 and Table 7 below. The distribution of the number of commercial and volunteer users is highly skewed toward the lower end of the range and can be approximated with a log-normal distribution. To a lesser degree the distributions of the number of community focused developers, product focused developers, and commits are also skewed. To accommodate for this in our models, the logarithm (base 2) of these variables is used. Of note regarding the correlations is the high correlation between the number of volunteer developers at time t and time $t-1$. This is a sign of a broader problem of autocorrelation in the number of volunteer developers across many time periods, which can be seen in Figure 1. This high level of autocorrelation can be compensated for by examining the difference in the number of volunteer developers between different time periods. The autocorrelation of this new variable is seen in Figure 2. The autocorrelations are now significantly lower, however there remains a moderately high autocorrelation with a time lag of one. This indicates that periods with high volunteer influx will often be followed by volunteer outflow in subsequent periods. We consider this correlation of -0.25 as sufficiently low and only have negligible impact on the regression models in Equation 1 and 2.

Insert Table 6 about here

Insert Table 7 about here

Insert Figure 1 about here

Insert Figure 2 about here

We begin with a regression model that predicts the change in the logarithm of the number of volunteer developers contributing source code to project at time t as a function of the log transformed number of volunteer developers, commercial developers, and commits at time $t-1$. To accommodate for the varying level of inherent attractiveness for different projects in the ecosystem, each project in the regression was represented by an indicator variable, α_i, to fit the so-called fixed-effects model. The regression model is shown in Equation 1. Within this model, the response variable is the difference in the log of the number of volunteer developers, which is functionally the same as the percentage change in the number of volunteer developers, for project i from time period $t-1$ to t, $\text{diff}\big(\log\big(VolDevs_{i,t}\big)\big)$ and the predictor variables are the intercept for the project, α_i, the log of the number of volunteer developers for project i in the previous time period, $\log\big(VolDevs_{i,t-1}\big)$, the log of the number of commercial developers for project i in the previous time period, $\log\big(ComDevs_{i,t-1}\big)$, the number of commits to the project version control system for project i in the previous time period, $\log\big(Commits_{i,t-1}\big)$, and an identifier for the current time period, $\log(t)$.

Equation 1:

$$\text{diff}\big(\mathbf{log}\big(\boldsymbol{VolDevs_{i,t}}\big)\big) = \boldsymbol{\alpha_i} + \boldsymbol{\beta_0}\,\mathbf{log}\big(\boldsymbol{VolDevs_{i,t-1}}\big) + \boldsymbol{\beta_1}\,\mathbf{log}\big(\boldsymbol{ComDevs_{i,t-1}}\big) + +\boldsymbol{\beta_2}\,\mathbf{log}\big(\boldsymbol{Commits_{i,t-1}}\big) + \boldsymbol{\beta_3}\,\mathbf{log}(\boldsymbol{t}) + \boldsymbol{\epsilon_{i,t}}$$

The results of the model, reported in Table 8 indicate that an increase in the number commercial software developers working on a project has no effect on attracting additional volunteer developers to the project. However, general activity in a project, as measured by the number of commits, is related to an increase in the number of volunteer developers in the next time period. This effect is tempered by the fact that project growth typically slows down relative to the size of the project as project ages, and therefore, they attract

fewer new volunteers relative to their existing size. Coefficients for project indicator variables, not shown, ranged from 0.09 to 0.89 and were significant at the $p < 0.001$ level for 11 of the 13 projects. Due to the lack of significance of $\log(ComDevs_{i,t-1})$ we conclude there is not support for hypothesis 1 or hypothesis 2.

Insert Table 8 about here

As we have shown there is a marked difference between the methods and magnitudes of participation of the two types of commercial developers: product focused and community focused. In Equation 2 we expand on the model through the use of firm type as a moderating variable to differentiate between participation of developers for community focused firms, $ComDevs_{CF,t-1}$, and product focused firms $ComDevs_{PF,t-1}$. The same data are used with the new multi-level longitudinal model, with the regression coefficients presented in Table 9.

Equation 2:

$$\begin{aligned}\text{diff}(\log(\boldsymbol{VolDevs_{i,t}})) &= \boldsymbol{\alpha_i} \\ &+ \boldsymbol{\beta_0}\log(\boldsymbol{VolDevs_{i,t-1}}) + \boldsymbol{\beta_1}\log(\boldsymbol{ComDevs_{CF_{i,t-1}}}) + \boldsymbol{\beta_2}\log(\boldsymbol{ComDevs_{PF_{i,t-1}}}) \\ &+ \boldsymbol{\beta_3}\log(\boldsymbol{Comits_{i,t-1}}) + \boldsymbol{\beta_4}\log(\boldsymbol{t_i}) + \boldsymbol{\epsilon_{i,t}}\end{aligned}$$

Insert Table 9 about here

In contrast to the original model where there was no significant relationship between firm participation and the change in the number of volunteers, when the firms are classified by business model, there is a significant difference. Participation by developers from community-focused firms has a significant and positive relationship to the change in the number of volunteer users, while participation by developers for product focused firms has no statistically significant impact. The other coefficients in the model remain similar to those shown for the more basic model shown in Table 8 and the explanatory power of the

model has increased slightly. This difference between developers at community and product focused firms lends support for hypothesis 3.

In order to evaluate Hypothesis 4, we needed to subdivide projects into their constituent modules, and use modules, rather than projects, as our unit of analysis. While language specific methods exist to specify explicit module structures and infer implicit structure through static source code analysis, the code in the GNOME project is written in a variety of different languages and programming language specific methods are inconsistent and impractical. As an alternative, we used social network analytic clustering methods on the network of source code to approximate modules within the project. While a variety of unsupervised clustering methods exist that heuristically determine an optimal number of clusters (e.g. Newman, 2004) these methods often produce more clusters than practical, leaving many clusters with only a single active developer. The CONCOR algorithm was used as it requires no additional information beyond link information when generating the groupings and groups elements based on structural similarity. Functionally, the computation views the network of files as a matrix and then attempts to rearrange the rows and columns of the matrix so entities that are structurally equivalent, meaning they link to the same set of other files, are grouped together (Boorman & White, 1976).

To obtain the network for CONCOR clustering, we utilized logical dependency links between files: nodes in the network are files, and edges are added between nodes if they were committed back to the central repository in a single commit. This approach is commonly used in software engineering research and is based on the observation that files are committed together by the same person frequently have dependencies on each other and is suitable for situations where language specific methods fail (Gall, Hajek, & Jazayeri, 1998). In this way, we obtain a network where highly related files are densely clustered together. The CONCOR algorithm was run on this network for each project and configured to generate 8 clusters, each approximating a module within the project. As CONCOR is a binary slicing

algorithm we needed to pre-specify the number of groups for each project – eight was chosen as a compromise for very large projects, for which four groups would result in many disparate modules being lumped together, and smaller projects, for which there were not sixteen different components. The same summary statistics shown in Table 6 were generated for each module in each eight week time period, yielding a total of 6360 observations.

The analysis at the project level was then replicated with the new data based on the clusters within each project. For this analysis, a new subscript j is added to the model indicating the cluster within project ii. Intercepts are calculated for each of the clusters in the data, $\alpha_{i,j}$, and time is the number of 8-week periods since the start of the project. The complete model is shown in Equation 3 and the results appear in Table 10.

Equation 3:

$$\begin{aligned}\text{diff}\left(\log\left(\boldsymbol{VolDevs_{i,j,t}}\right)\right) &= \boldsymbol{\alpha_{i,j}} \\ &+ \boldsymbol{\beta_0}\log\left(\boldsymbol{VolDevs_{i,j,t-1}}\right) \\ &+ \boldsymbol{\beta_1}\log\left(\boldsymbol{ComDevs_{i,j,t-1}}\right) + \boldsymbol{\beta_2}\log\left(\boldsymbol{Commits_{i,j,t-1}}\right) + \boldsymbol{\beta_3}\log(\boldsymbol{t_i}) + \boldsymbol{\epsilon_{i,j,t}}\end{aligned}$$

Insert Table 10 about here

The effects of the model largely mirror the results found when analyzing at the project level (see Table 8). The lack of significance for the coefficient of $\log\left(ComDevs_{i,j,t-1}\right)$ shows a lack of support for Hypothesis 4. As before, we examine the difference in the effect of commercial developers when the developers are segregated by firm. This differentiation results in a new model shown in Equation 4 and the results are presented in Table 11.

Equation 4:

$$\begin{aligned}\text{diff}\left(\log\left(\boldsymbol{VolDevs}_{i,j,t}\right)\right) &= \boldsymbol{\alpha}_{i,j} \\ &+ \boldsymbol{\beta_0}\log\left(\boldsymbol{VolDevs}_{i,j,t-1}\right) \\ &+ \boldsymbol{\beta_1}\log\left(\boldsymbol{ComDevs}_{\boldsymbol{CF}_{i,j,t-1}}\right) \\ &+ \boldsymbol{\beta_2}\log\left(\boldsymbol{ComDevs}_{\boldsymbol{PF}_{i,j,t-1}}\right) + \boldsymbol{\beta_3}\log\left(\boldsymbol{Commits}_{i,j,t-1}\right) + \boldsymbol{\beta_4}\log(\boldsymbol{t_i}) + \boldsymbol{\epsilon_{i,j,t}}\end{aligned}$$

Insert Table 11 about here

Under this new model we see that at the module level community focused developers once again are related to an increase in volunteer participation and that product focused developers have a significant negative relation to volunteer participation. This paints a mixed picture regarding the increase in cognitive complexity at the module level. For product focused developers there is support for hypothesis 4. However, community focused developers retain their positive relationship with the change in the number of volunteer developers, indicating that whatever cognitive complexity they introduce may be overcome by their general power to attract new developers to the community. Therefore, we reject hypothesis 4 for community focused developers.

DISCUSSION

As Open Source continues to grow, more commercial developers will need to find ways to work with pre-existing communities of volunteers and other commercial developers. While there are numerous ways that commercial developers can affect the level of volunteer and independent participation in these projects, this work establishes that in terms of the number of volunteer developers, there is little overall negative impact from commercial developers participating in existing Open Source communities. At the aggregate we observe that commercial developers have no statistically significant impact on the change in volunteer developers, which does not allow us to confirm or reject hypotheses 1 and 2. For Open Source community managers and project managers, this should be good news as it indicates that their population

of volunteers will most likely remain steady in the presence of commercial interests utilizing and modifying the project code for commercial purposes.

Another major contribution is the identification and validation of two distinct classes of business models for firms participating in Open Source communities and the different impact they have on volunteer participation. These models closely aligned themselves either with broad community success or with a single project in the community. As expected, community focused firms were found to be highly visible on project mailing lists and wrote code for more projects than developers from product focused firms. Yet, contrary to our assumptions, product focused firms were more active in the moderately technical domain of bug reporting and management. The different business models were found to have very different impacts on volunteer developers – with community focused developers attracting volunteers while product focused developers had no statistically significant effect, supporting hypothesis 3. Given the predominant view of the interviewees that community focused firms were more aligned with the values and norms of the community, this supports the notion that the communities are sensitive to the values and norms of commercial participations. This result should urge caution on firms wishing to participate in Open Source projects, and suggests that behaving in a way that supports the community may actually strengthen and enlarge it.

Finally, we analyzed whether or not commercial developers had a negative effect on volunteer participation at the module level, possibly due to some of the properties of commercial development such as developer collocation, increased ability to work with others, or increased and technical skill. Much to our surprise it was found that this was the case for developers from product focused firms, but not community focused firms. This indicates that there is the possibility the commercial firms working on a module within a large Open Source project may drive volunteer developers away, as is the case with product focused firms. However, this doesn't need to be the case, the differences in behavior for

community focused developers along with their general attractive power seem to negate these negative effects and once again show are correlated with an increase in the number of volunteer developers.

Our research suggests both caution and some reassurance for firms considering a product-focused relationship to an Open Source community. Our qualitative results show that volunteer developers frequently made negative comments about product focused firms, which is quite worrisome. On the other hand, increased participation of developers from product-oriented firms did not drive volunteers away. It may be that the increased visibility that commercial participation lends a project offsets any negative effects from its perceived failure to uphold community norms.

No matter what the reasons for the increased success of community focused firms in attracting and retaining volunteer developers, it appears that more and more firms will adopt Open Source projects, practices, and contribute to communities in the near future. We have seen that in this case, the dual worlds of volunteer and commercial can co-exist in an Open Source project with little danger of the commercial firm dramatically damaging the incumbent volunteers. Going forward, understanding the methods by which these firms attract and retain volunteer developers is an open research question that will yield great benefits for firms seeking to utilize this revolutionary software development model.

BIBLIOGRAPHY

Ashforth, B. E., & Mael, F. 1989. Social Identity Theory and the Organization. ***The Academy of Management Review***, *14*(1): 20-39.

Baldwin, C. Y., & Clark, K. B. 2000. ***Design Rules, Vol. 1: The Power of Modularity*** (First Edition.), Cambridge, MA: The MIT Press.

Bartholomew, D. 2008. A look at the Kindle. ***Linux Journal***, *2008*(176): 3.

Barton, J. 2003. From Server Room to Living Room. ***Queue***, *1*(5): 20-32.

Bitzer, J., Schrettl, W., & Schroder, P. J. 2007. Intrinsic motivation in open source software development. ***Journal of Comparative Economics***, *35*(1): 160-169.

Bonaccorsi, A., Giannangeli, S., & Rossi, C. 2006. Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry. ***Management Science***, *52*(7): 1085-1098.

Boorman, S. A., & White, H. C. 1976. Social Structure from Multiple Networks. II. Role Structures. ***American Journal of Sociology***, *81*(6): 1384-1446.

Brooks, F. P. 1995. ***The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition*** (2nd ed.), Boston, MA, USA: Addison-Wesley Professional.

Campagnolo, B., Tacla, C. A., Paraiso, E. C., Sato, G. Y., & Ramos, M. P. 2009. An architecture for supporting small collocated teams in cooperative software development. In ***International Conference on Computer Supported Cooperative Work in Design***: 264-269, Los Alamitos, CA, USA: IEEE Computer Society.

Fielding, R. T. 2005. Software architecture in an open source world. In ***27th International Conference on Software Engineering***: 43, Presented at the 2005 International Conference on Software Engineering (ICSE 2005), St. Louis, MO, USA.

Fielding, R. T. 1999. Shared leadership in the Apache project. ***Communications of the ACM***, *42*(4): 42-

43.

Fogel, K. 2005. ***Producing Open Source Software***, Sebastapol, CA: O'Reilly & Associates.

Gall, H., Hajek, K., & Jazayeri, M. 1998. Detection of Logical Coupling Based on Product Release History. In ***14th IEEE International Conference on Software Maintenance***, Presented at the International Conference on Software Maintenance, Bethesda, MD: IEEE Press.

German, D. 2005. Software Engineering Practices in the GNOME Project. In J. Feller, B. Fitzgerald, S. A. Hissam, K. R. Lakhani, & M. Cusumano (Eds.), ***Perspectives on Free and Open Source Software***: 211-226, Cambridge, MA: MIT Press.

German, D. 2004. The GNOME project: a case study of open source, global software development. ***Software Process: Improvement and Practice***, *8*(4): 201-215.

Ghosh, R. A., Glott, R., Krieger, B., & Robles, G. 2002. ***Free/Libre and Open Source Software: Survey and Study***, International Institute of Infonomics University of Maastricht, The Netherlands.

Goldman, R., & Gabriel, R. P. 2005. ***Innovation Happens Elsewhere: Open Source as Business Strategy***, San Francisco, CA: Morgan Kaufmann.

Hars, A., & Ou, S. 2002. Working for Free? Motivations for Participating in Open-Source Projects. ***International Journal of Electronic Commerce***, *6*(3): 25-39.

Hertel, G., Niedner, S., & Hermann, S. 2003. Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel. ***Research Policy***, *32*(7): 1159-1177.

de Icaza, M. 2002, April 9. The Story of the GNOME Project, http://primates.ximian.com/~miguel/gnome-history.html, January 10, 2010.

Jackson, S. E., Brett, J. F., Sessa, V. I., Cooper, D. M., Julin, J. A., & Peyronnin, K. 1991. Some differences make a difference: Individual dissimilarity and group heterogeneity as correlates of recruitment, promotions, and turnover. ***Journal of Applied Psychology***, *76*(5): 675-689.

Koch, S., & Schneider, G. 2002. Effort, co-operation and co-ordination in an open source software

project: GNOME. ***Information Systems Journal***, *12*(1): 27-42.

Krishnamurthy, S. 2005. An Analysis of Open Source Business Models. In J. Feller, B. Fitzgerald, S. Hissam, & K. R. Lakhani (Eds.), ***Perspectives on Free and Open Source Software***, Cambridge, MA: MIT Press.

Lakhani, K., & Wolf, R. 2005. Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. In Joseph Feller, Brian Fitzgerald, Scott Hissam, & Karim R. Lakhani (Eds.), ***Perspectives on Free and Open Source Software***, Cambridge, MA: MIT Press.

Lawton, G. 2008. US Cell Phone Industry Faces an Open Future. ***Computer***, *41*(2): 15-18.

MacCormack, A., Rusnak, J., & Baldwin, C. Y. 2006. Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. ***Management Science***, *52*(7): 1015-1030.

Maxwell, E. 2006. Open Standards, Open Source, and Open Innovation: Harnessing the Benefits of Openness. ***Innovations: Technology, Governance, Globalization***, *1*(3): 119-176.

Moody, G. 2001. ***Rebel Code: Linux and the Open Source Revolution***, New York, NY: Basic Books.

Neus, A., & Scherf, P. 2005. Opening minds: Cultural change with the introduction of open-source collaboration methods. ***IBM Systems Journal***, *44*(2): 215-225.

Newman, M. 2004. Detecting community structure in networks. ***The European Physical Journal B***, *38*(2): 321-330.

Oreg, S., & Nov, O. 2008. Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values. ***Computers in Human Behavior***, *24*(5): 2055-2073.

Parnas, D. 1972. On the Criteria to be Used in Decomposing Systems Into Modules. ***Communications of the ACM***, *15*(12): 1053-1058.

Pelled, L. H., Eisenhardt, K. M., & Xin, K. R. 1999. Exploring the Black Box: An Analysis of Work Group Diversity, Conflict, and Performance. ***Administrative Science Quarterly***, *44*(1): 1-28.

Polletta, F., & Jasper, J. M. 2001. Collective Identity and Social Movements. ***Annual Review of Sociology***, *27*(1): 283-305.

Raymond, E. S. 1999. ***The Cathedral and the Bazaar***, Sebastapol, CA: O'Reilly & Associates.

Roberts, J. A., Hann, I., & Slaughter, S. A. 2006. Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. ***Management Science***, *52*(7): 984-999.

Rosen, L. 2004. ***Open Source Licensing: Software Freedom and Intellectual Property Law***, Upper Saddle River, NJ: Prentice Hall PTR.

Shah, S. K. 2006. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. ***Management Science***, *52*(7): 1000-1014.

Star, S. L., & Griesemer, J. R. 1989. Institutional Ecology, `Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. ***Social Studies of Science***, *19*(3): 387-420.

Stewart, K. J., & Gosain, S. 2006. The Impact of Ideology on Effectiveness in Open Source Software Development Teams. ***MIS Quarterly***, *30*(2): 291-314.

The GNOME Foundation. 2010, January. GNOME: The Free Software Desktop Project, http://www.gnome.org/, January 6, 2010.

Tiemann, M. 1999. Future of Cygnus Solutions: An Entrepreneur's Account. In C. DiBona, S. Ockman, & M. Stone (Eds.), ***Open Sources: Voices from the Open Source Revolution***: 71-91, Sebastapol, CA: O'Reilly Media, Inc.

Williams, K., & O'Reilly, C. 1998. Demography and diversity in organizations: A review of 40 years of research. ***Research in Organizational Behavior***, *20*: 77-140.

Table 1: General Description of Interviewees

Total Interviewees	18
Commercial Developers	50%
Volunteer Developers	50%
Student Volunteer Developers	17%
Commit Access	78%
Self-Described as Developer	89%
Self-Described as Community Support	11%
Longest Participation	10 years
Shortest Participation	11 months
Median Participation	3 years

Table 2: Major firms participating in the community

Product Focused Firms		Community Focused Firms	
Firm A	A large IT firm that became involved through the purchase of Firm B and has greatly expanded to a wide variety of desktop applications and a distribution of Linux.	Firm F	A large Linux distributor that distributes GNOME and sells and supports Linux for the desktop and server.
Firm B	A medium firm that developed enterprise class desktop applications and system support for the GNOME. Purchased by Firm A.	Firm G	A large IT firm that uses GNOME as a compliment to its hardware offerings.
Firm C	A small firm that specializes in adapting small portions GNOME for embedded applications.	Firm H	A Linux distributor that historically shipped a desktop environment from a competing Open Source community and had small participation in the community.
Firm D	A small firm that produced enterprise class desktop applications for the GNOME community.	Firm I	A medium Linux distributor that historically supported both GNOME and another community with a similar goal.
Firm E	A small venture capital funded firm that developed software and sold integrated services for the GNOME community		

Table 3: Mean Activity per Year on Mailing Lists by Class of Developer (superscripts indicate statistically different groups of means in each row)

Variable	Volunteer Mean	Product Focused Mean	Community Focused Mean	P Value
Messages	39.04^{A}	87.10^{B}	135.80^{C}	<0.001
Threads Started	13.85^{A}	34.42^{B}	56.37^{B}	<0.001
Mailing Lists	0.37^{A}	0.68^{B}	0.53^{B}	<0.001
Extra Mailing Lists	0.20^{A}	0.23^{A}	0.20^{A}	0.400
Email References/Message	0.22^{A}	0.22^{A}	0.41^{B}	<0.001
URLs/Message	0.48^{A}	0.31^{B}	0.81^{C}	<0.001

Table 4: Mean Activity per Year in Bug Reporting Database by Class of Developer (superscripts indicate statistically different groups of means in each row)

Variable	Volunteer Mean	Product Focused Mean	Community Focused Mean	P Value
Comments	74.92^{A}	156.50^{B}	133.40^{B}	0.010
Patches	4.93^{A}	9.60^{A}	6.14^{A}	0.042
Bugs Fixed	1.44^{A}	3.80^{B}	7.30^{B}	<0.001
Projects	2.60^{A}	3.54^{B}	2.43^{A}	0.141
Extra Projects	1.56^{A}	2.09^{B}	1.11^{A}	0.556

Table 5: Mean Activity per Year in CVS Repository by Class of Developer (superscripts indicate statistically different groups of means in each row)

Variable	Volunteer Mean	Product Focused Mean	Community Focused Mean	P Value
CVS Projects	13.72^{A}	5.24^{B}	15.29^{A}	0.002

Table 6: Summary Statistics of Data Collected from 14 projects at 8 week intervals (601 total observations)

	Mean	Median	Skewness	Kurtosis	Std Dev	Max	Min
$VolDevs_{i,t}$	4.01	3	1.24	0.96	3.94	18	0
Number of volunteer developers contributing code to project i at time t							
$ComDevs_{i,t}$	3.57	2	2.19	4.70	4.87	26	0
Number of commercial developers contributing code to project i at time t							
$ComDevs_{CF,i,t}$	1.12	1	2.16	4.77	1.66	9	0
Number of commercial developers from community focused firms contributing code to project i at time t							
$ComDevs_{PF,i,t}$	2.65	1	2.84	8.22	4.49	26	0
Number of commercial developers from product focused firms contributing code to project i at time t							
$Commits_{i,t}$	114.97	46	2.98	11.54	175.64	1407	0
Number of commits made by all developers to project i at time t							
Observations per Project, N	42.92	49	-1.66	1.74	12.72	53	14

Table 7: Correlations of Data Collected at Project Level after $\log_2$ transformations.

	$VolDevs_t$	$VolDevs_{t-1}$	$ComDevs_{t-1}$	$ComDevs_{CF,t-1}$	$ComDevs_{PF,t-1}$
$VolDevs_{t-1}$	0.8263				
$ComDevs_{t-1}$	0.3755	0.3921			
$ComDevs_{CF,t-1}$	0.4346	0.4258	0.6272		
$ComDevs_{PF,t-1}$	0.2733	0.2773	0.9272	0.3555	
$Commits_{t-1}$	0.6655	0.7331	0.6400	0.4208	0.5504

Table 8: Hypothesis 1 and 2 – Regression Coefficients Predicting Change in Number of Volunteer Developers by Project (Equation 1)

Variable	Estimate	Std Err	P Value
$\log(VolDevs_{i,t-1})$	-0.4779	0.0356	<.001
$\log(ComDevs_{i,t-1})$	0.0411	0.0311	0.187
$\log(Commits_{i,t-1})$	0.0819	0.0199	<.001
$\log(t_i)$	-0.0471	0.0156	0.003
R^2=0.211, Adj R^2=0.193, DF=746, $p < 0.0001$			

Table 9: Hypothesis 3 – Multi-Level Regression Coefficients Predicting Number of Volunteer Developers by Project Classified By Firm Model (Equation 2)

Variable	Estimate	Std Err	P Value
$\log(VolDevs_{i,t-1})$	-0.4871	0.0356	<.001
$\log(VolDevs_{CF_{i,t-1}})$	0.0918	0.0350	0.008
$\log(VolDevs_{PF_{i,t-1}})$	-0.0212	0.0299	0.479
$\log(Commits_{i,t-1})$	0.0843	0.0194	<.001
$\log(t_i)$	-0.0401	0.0159	0.011
R^2=0.217, Adj R^2=0.198, DF=745, $p < 0.0001$			

Table 10: Hypothesis 4 – Testing for issues of cognitive complexity through the analysis of effect of commercial developers at the module level (Equation 3)

Variable	Estimate	Std Err	P Value
$\log(VolDevs_{i,j,t-1})$	-0.5405	0.0142	<.001
$\log(ComDevs_{i,j,t-1})$	0.0038	0.0132	0.774
$\log(Commits_{i,j,t-1})$	0.0992	0.0079	<.001
$\log(t_i)$	-0.0022	0.0055	0.693
$R^2 = 0.226$, Adj R^2=0.213, $DF = 5996$, $p < 0.0001$			

Table 11: Hypothesis 4 – Testing for issues of cognitive complexity through the analysis of effect of commercial developers at the module level (Equation 4)

Variable	Estimate	Std Err	P Value
$\log(VolDevs_{i,j,t-1})$	-0.5464	0.0142	<.001
$\log\left(ComDevs_{CF_{i,j,t-1}}\right)$	0.0616	0.0156	<.001
$\log\left(ComDevs_{PF_{i,j,t-1}}\right)$	-0.0284	0.0134	0.034
$\log(Commits_{i,j,t-1})$	0.0996	0.0075	<.001
$\log(t_i)$	0.0004	0.0055	0.934
$R^2 = 0.229$, Adj R^2=0.215, $DF = 5995$, $p < 0.0001$			

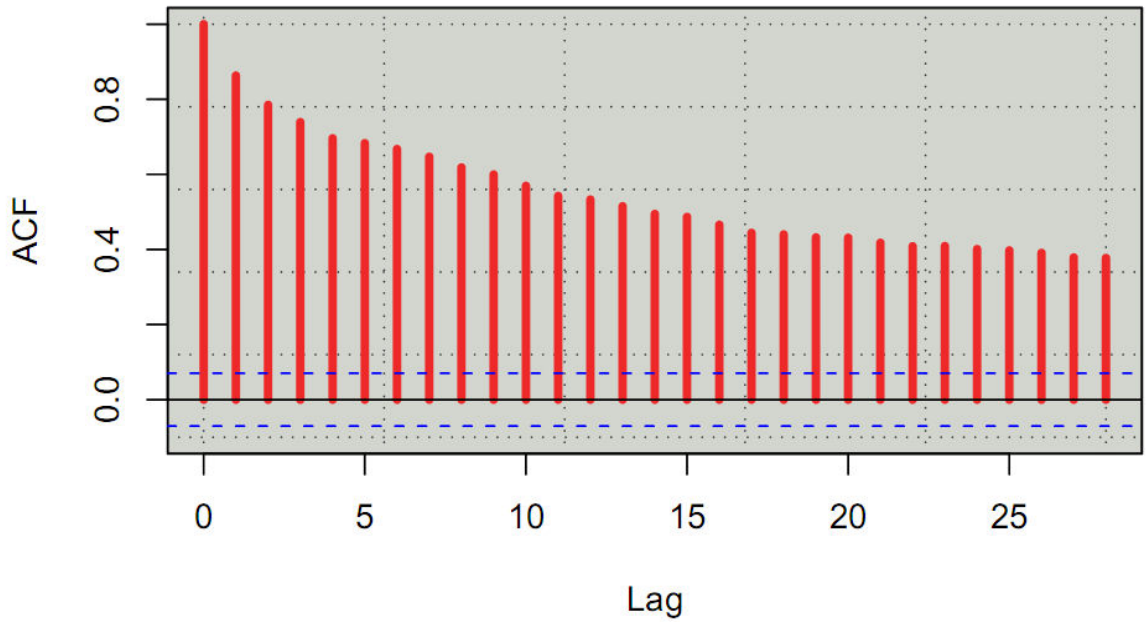

Figure 1: Autocorrelation between time periods of the number of volunteer developers in the GNOME community. One time unit represents eight weeks of calendar time.

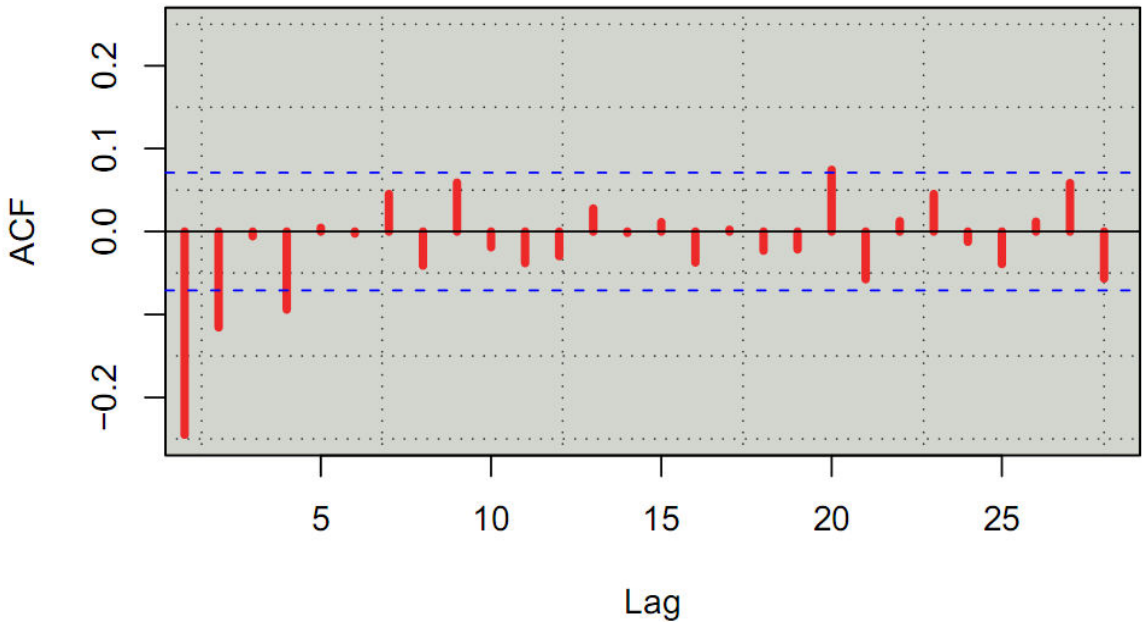

Figure 2: Autocorrelation between time periods of the change in the number of volunteer developers in the GNOME community. One time unit represents eight weeks of calendar time.