Language and Power in Self-organizing Distributed Teams

ABSTRACT

In this paper, a comparative case study is conducted to explore the way power is expressed and exercised through language use in distributed or virtual teams. Our research questions are "how is power expressed in online interactions in self-organizing distributed teams, in a context without formal authority or hierarchy?" and "What effects do expressions of power have on team outcomes?" To fully understand the role of power in self-organizing teams, we apply an input-process-output model on two open source projects-one successful and the other less successful. Two set of codes (source of power and power mechanism) are drawn from the data, and different power patterns interestingly show up between them. The findings lead us to speculate that strong, centralized leadership, the assertive exercise of power, and direct language may contribute to effectiveness in FLOSS teams. And the relevant conclusions and suggestions are provided for further research.

Keywords:

Power; Leadership; Distributed Team, Language; FLOSS

Language and Power in Self-organizing Distributed Teams

1 Introduction

In this paper, we explore the way power is expressed through language use in distributed or virtual teams. We focus in particular on self-organizing teams in which the teams' operations do not depend on formal authority, roles or rules assigned by outside institutions. Such teams are growing in importance as organizations seek the flexibility offered by distributed teams and in particular, the ability to quickly form teams including participants from multiple organizations. However, few scholars have studied the emergence and role of power in self-organizing teams. One reason for this omission may be that traditional notions of power and leadership do not seem easy to apply to such teams. At the risk of oversimplifying, most leadership theories have tended to view the leader as a single, dominant individual (the "great man"), usually (though not always) occupying a formally defined leadership position in the social structure. These approaches do not seem to provide much leverage for understanding leadership and power in self-organizing distributed teams. Such teams are often composed of people of relatively equal status, or who are so disparate in background that formal organizational status seems irrelevant, reducing the usual leadership cues provided by organizational status and title. Indeed, they often have no appointed leader, and their members may or may not have significant prior experience working with one another. In such cases, rather than being appointed or even elected, a leader or leaders may emerge gradually, and such emergent leadership may be completely unrelated to organizational position or status.

Nevertheless, the distributions of power and influence mechanism seem likely to be related to the effectiveness of organizations, making them important to study. Kanter (1979) notes that executive and managerial power is a necessary ingredient for moving organizations toward their goals: "Power can mean efficacy and capacity" for organizations. Therefore, in this paper, we explore the role of power in self-organizing distributed groups. Our research questions are “How is power expressed in online interactions in self-organizing distributed teams?” and “What effects do expression of power have on team outcomes?”

1.1 Research setting

Our study is based on a case study of two Free/Libre Open Source Software (FLOSS) projects. FLOSS[1] is a broad term used to embrace software developed and released under an “open source” license allowing inspection, modification and redistribution of the software’s source code. There are thousands of FLOSS projects, spanning a wide range of applications. Due to their size, success and influence, the Linux operating system and the Apache Web Server are the most well known, but hundreds of others are in widespread use, including projects on Internet infrastructure (e.g., sendmail, bind), user applications (e.g., Mozilla, OpenOffice) and programming languages (e.g., Perl, Python, gcc).

FLOSS projects are prime examples of self-organizing distributed teams. Developers contribute from around the world, meet face-to-face infrequently (sometimes not at all) and coordinate their activity primarily by means of computer-mediated communications (CMC) (Raymond, 1998; Wayner, 2000). The teams have a high isolation index (O’Leary & Cummings, 2002) in that most team members work on their own and in most cases for different organizations (or no organization at all). As a result, these teams depend on processes that span traditional boundaries of place and ownership (Watson-Manheim, Crowston & Chudoba, 2002). The research literature on software development and on distributed work emphasizes the difficulties of distributed software development, but the case of FLOSS development presents an intriguing counter-example.

[1] FLOSS software is generally available without charge (“free as in beer”). Some (though not all) OSS software is also “free software”, meaning that derivative works must be made available under the same license terms (“free as in speech”, thus “libre”). We have chosen to use the acronym FLOSS rather than the more common OSS to accommodate this range of meanings.

FLOSS teams have several features that set them apart from the distributed teams that have been studied in prior research. First, many (though by no means all) programmers contribute to projects as volunteers, without working for a common organization or being paid. Second, in addition to intra-group management, non-member involvement plays an important role in the success of the teams. Users who are non-members or peripheral members contribute to the projects in multiple ways, and become a crucial resource of potential recruitment (Heckman, Li & Xiao, 2006). How to handle the relationship between non-members' requirements and the project goal is also a big challenge. Finally, the teams are largely self-organizing, often without formally appointed leaders or indications of rank or role. Though core group membership can bestow some rights, including deciding what features should be integrated in the release of the software, when and how to empower other code maintainers, or to "pass the baton" to the next volunteer (Raymond 1999), in comparison to traditional organizations, more people can share power and be involved in group activities. In many projects, anyone with enough interest and skills can access the code, contribute patches, make suggestions to group, and attend important decision processes. These features make FLOSS teams extreme examples of self organizing distributed teams, but they are not inconsistent with what many organizations are facing in recruiting and motivating professionals and in developing distributed teams. As Drucker has said, increasingly knowledge workers have to be managed as if they were volunteers. These characteristics of self organization and volunteerism make the question of power particularly problematic in FLOSS teams.

In the following sections of the paper, we first briefly review the extensive literature on power, focusing specifically on a definition of power, the sources of power in traditional organizations, and power-related research in the context of Free/libre Open Source Software (FLOSS) development teams. We then present a comparative analysis of two FLOSS projects. Interestingly, we found that the project with a dominating and impolite leadership style seems to have obtained better outcomes than the project with a more democratic and polite style. This finding is explored though an in-depth interpretive analysis of interaction episodes on the developer's email list of the two projects, which provide valuable explanations regarding the emergence of power and the role of power in project outcomes. The last section of the paper briefly discusses the limitation and implications of the study and provides suggestions for further research.

2 Theory: Power and language

Power has been recognized as an important and useful construct in formal or informal organization studies for several decades (Pfeffer, 1981; Allen and Porter, 1983; Shafritz and Steven 1987). However, power is a "messy, elusive concept that not only has surface or visible characteristics, but also hidden characteristics that are difficult to define and grasp" (Jasperson, Carte, Saunders, Butler, Croes & Zheng, 2002). Multiple concepts such as authority, polities or influence have been used in much the same way as power. However, no matter what the definition, it usually means that a certain social actor possesses a capability to get others to follow his will in achieving a desired objective (Kanter 1979; Mechanic 1962; Allen and Potter 1983; Jasperson et al 2002; Higgins, Judge & Ferris, 2003). Here, we follow Shafritz and Steven (1987)'s definition: "Power is the ability to get things done, the way one wants them done; it is the latent ability to influence people."

To develop a model of the role of power in self-organizing teams, we apply an input-process-output model. It not only allows us to consider antecedents and outcomes, but also allows us to examine the intervening processes in detail and understand power mechanism systematically. For inputs, we consider the antecedents of power, i.e., where power comes from. It is important to note that power is context or relationship-specific. In other words, we can not say a person is powerful or powerless in general; rather power is determined by the situation and the relationships of one individual with others (Pfeffer 1977). Generally, power is conceived of as deriving from two sources. First, power comes from formal authority, which is often held by higher-ranked participants in organizations. This kind of power stems from legitimate authority (authority flowing down through organizational hierarchy) and formal rules (promulgated and enforced by those in authority) to ensure that organizational behavior is directed toward the attainment of established organizational goals (Shafritz and Ott, 1987). Such power might also be the product of a formal decree (Astley and Sachdeva, 1984). For self-organizing teams, these sources of power are less relevant, as there are likely no authorities recognized by all participants. The second source of power goes beyond formal authority to sources including expertise, effort and interest, attractiveness, location and position, coalitions and rules (Mechanic 1962) by which participants can make others dependent on them and thus obtain power or influence. Jasperson et al. (2002) found that ICT use can lead to greater equality of participation in decision making and thus reduce the salience of influence behavior by giving more individuals and coalitions an opportunity to have their positions heard.

To examine the process of power, we draw on literature on power and language, which hypothesizes that power differences between individuals will be reflected in their organizational interactions and behaviors, in particular, in the style of communication. Understanding communication style therefore will help us understand underlying power differences and the mechanisms through which it is expressed. For example, Morand (1996) explored how power is expressed in the speech interactions between superiors and subordinates. He drew upon politeness theory, which describes the linguistic behaviors used to demonstrate regard and consideration for others. He found in a lab study that use of politeness is sensitive to the distribution of power. Low power actors were most likely to use linguistic politeness behaviors to minimize the possibility of conflict with superiors. Rogers and Lee-Wong (2003) developed a framework explaining the tensions that occur as subordinates attempt to maintain a sufficient degree of politeness while reporting to superiors on workplace tasks.

Finally for the output of power, our overall concern is the effects of power use on the effectiveness of FLOSS teams. We assess effectiveness in a variety of ways, following the model of Crowston et al (2005), as discussed below.

2.1 Power in FLOSS

As noted above, there has been little research about power issues in the FLOSS context. Due to the relationship between power and leadership, the few papers we have mostly touch upon power derived from the role of leader. Brian et al. (2002) found that the owners (or administrator, host or wizard) of a project are typically assigned special email addresses, are prominently identified in the description of the community, and have special privileges (add or remove members from community, or remove items from archives etc). Bradner (2003) described one source of power as, "if someone is seen as strong and as having done good work in the past (and can argue their position well) they gain quite a bit of status and future proposals from them tend to get a better ride."

Besides understanding the characteristics of leader-based power, some papers went further on studying how the power might exert a positive influence on FLOSS projects. Lee and Waguespack (2005) found that open innovation communities such as FLOSS teams rely heavily upon strong leadership to function effectively and resist forking. They stated "because project members cannot be forced to participate in any activity or to pay attention to any other member, these relationships rely upon the power to persuade". In a case study of Linux community, Wendel et al. (2001) proposed the concept of self-organizing power and showed that apparent success of emergent ('bottom up') strategy finding processes ensures the interaction between change trajectories of both internal resource and external threats or weakness.

However, power might have negative outcomes if used improperly in certain contexts. Anthony (2003) discussed authority and control in firm involvement in FLOSS projects. He founded that the first and most significant change firms make to FLOSS projects is to the authority structure by controlling decisions such as access to mailing lists, which code contributions are accepted and combined into the project and who should be empowered to be subproject leader or succeed the current leaders. As a result, volunteers may feel detached from open-source projects because they are not included in the decision making process.

3 Research methodology

In this section, we discuss the research design, the two case sites, the data collection and analysis approach.

3.1 Research design

The study begins with the research question: "how is power expressed in online interactions in distributed teams?" and "what effects do the expressions of power have on team outcomes?" In order to answer to these questions, we adopted a comparative case study design. According to Feagin et al. (1991), case study is an ideal methodology when a holistic, in-depth investigation is needed. Power is an elusive construct that can not be identified and quantified easily, and is difficult to observe apart from the context in which it is exercised. Thus it is best investigated through holistic, in-depth observations carried out in a naturalistic setting. Because the case study is a multi-perspective analysis, we not only take the opinions and perspectives of the targeted actors or speakers into account, but also consider the actors in relevant groups (including voiceless people) and the interactions between them. This aspect is a salient point for situations such as FLOSS teams that contain both core and peripheral groups of actors.

Two case sites

Our study compares interactions in two comparable projects. Both are instant messenger applications that support multiple protocols. Gaim (http://sourceforge.net/projects/gaim/) is a multi-platform client, while Fire (http://sourceforge.net/projects/fire/) is specific to Mac OS X.

And both of them are community-based projects. Gaim was originally started by Mark A. Spencer in November of 1998[2]. But it began to be led to success by 1st-generation leader Rob Flynn who joined the team in the same year due to the desire for a good instant messaging program for Linux. Rob Flynn began to fade out from his leadership position gradually since 2002, when his successor Sean Egan was made an official part of the team with the frequent and valuable patches. Gaim released its Version 1.0.0 in 2004 and continued actively. Fire was founded by Eric Peyton in 2001[3]. He decided that because he needed an IM client which could run on the new OS from Apple and no official clients from IM vendors worked on it at that time. Eric is still the leader of Fire currently. Its Version1.0 was also released in 2004 and Fire keeps releasing regularly now.

We choose these two open source projects for two reasons. First, they are very similar and comparable on several dimensions. A comparison of the products, development status, licenses etc are presented in Table 1.

Insert Table 1 about here

Both projects are essentially successful: they created a community of developers and released functioning products that were used. However, drawing on the measurement of project success (Crowston, Howison &Annabi (2005) and the statistics made available by the FLOSSmole project[4] (Howison, Conklin & Crowston 2005), it is clear that Gaim is a more successful project than Fire. We operationalize success for this comparison along four criteria (Table 2): a) use, measured by downloads and popularity; b) activity, measured by online activity

[2] Project of the Month: Octorber 2002-Gaim. http://sourceforge.net/potm/potm-2002-10.php

[3] Fire homepage. http://fire.sourceforge.net/about.php

[4] FLOSSmole: http://ossmole.sf.net

frequency; c) spin-off projects or ports to other platforms; d) the ability to attract and retain developers and active users.

Insert Table 2 about here

From Figure 1, Gaim has substantially more downloads and pageviews than Fire, using statistics gathered by Sourceforge which both projects use as their main download site. Although Crowston et al (2005) caution that download and pageview figures should be adjusted for potential market size, in this case, the potential market sizes was initially essentially the same; the products performed the same function, both products were begun on a small market platform (Fire on Apple's OS X, Gaim on Linux with Gnome.). The later ports of Gaim to Windows and OS X (via the Darwin Ports system) discussed below, does increase the potential market size but this reflects a different dimension of success.

Insert Figure 1 about here

Gaim also dominates in terms of Activity, here represented by the Sourceforge activity percentile score and message count over time for the lists analyzed. Figure 2 shows the Activity percentile over time, both were often very high relative to the bulk of Sourceforge projects, but Gaim routinely was in the very highest percentile, while Fire never achieved that and varied down by 6% even after achieving project momentum.

Insert Figure 2 about here

Figure 3 shows that Gaim has both higher and more stable message counts.

Insert Figure 3 about here

On the third measure proposed in Crowston et al (2005), Gaim is again clearly dominant. It has spun off *libgaim*, which is the foundation of most cross-protocol clients and is now totally cross- platform. Gaim has also achieved a direct Windows port as well as a port to Apple's OS X via the Darwin ports system, both of which are reflected in Gaim's substantially higher download figures.

Finally, the projects differ strongly on their ability to attract and retain developers. In measurements over their approx five-year lifetime, Gaim has constantly and consistently attracted developers; while Fire initially attracted many initially but since then has lost developers. Figure 4 shows the size of the team given develop status in the Sourceforge system. This is not an ideal measurement of developer counts because the project's policies in granting such listing are unknown; it would be useful to bolster this analysis with an analysis based on the contribution of code. Nonetheless Gaim's ability to attract developers not only for its regular client, but also for a Windows and library port is clear.

Insert Figure 4 about here

3.2 Data collection

Data was gathered by collecting online interactions from their main developer mailing list on Sourceforge (Gaim: http://sourceforge.net/mailarchive/forum.php?forum_id=9587; Fire: http://sourceforge.net/mailarchive/forum.php?forum_id=9790). We choose these interactions because they are the communications between developers used to coordinate work, exchanges ideas and make group decisions. The users also tend to ask for help and involve in the group activities from developer mailing lists.

3.3 Analysis

To analyze the email interactions, we applied a qualitative inductive analysis technique. We began by examining forum messages to identify text segments referring to behaviors in which we expected to see power expressed, such as conflict management, decision-making, problem-solving and non-work issues. These segments were then assigned to theoretically meaningful categories derived initially from the literature review summarized above. However, the categories evolved through the course of the data analysis. As we coded each segment, we decided whether the segment fit an existing code, required a new code or required revision of the existing codes. We continued to revise the codes until each segment fit cleanly within some category. These codes were then grouped into higher-level categories and the relationships between these codes elaborated. The process resulted in two main sets of codes for sources of power and power mechanisms.

4 Findings

In this section, we summarize our findings along the dimensions discussed above.

4.1 Source of power

As we mentioned before, most of FLOSS projects are run without a central organizational authority in control. People contribute freely and voluntarily to the community as coding, providing solutions, or exchanging ideas with others. Previous studies on individual motivations show that people participate mainly based on intrinsic motivations, e.g. needing certain software, looking for fun, learning, giving back to community etc (Bitzer et al, 2004; Bonaccorsi and Rossi, 2003; Lakhani and Wolf, 2003). Due to these characteristics, power emerges very differently compared to traditional organizations where the nominal titles will bring much authority and social influence.

Based on our observations of two cases, four common sources of power were identified: expertise, effort and interest, role and control of the commit privileges.

4.1.1 Expertise

Expertise has been identified as an important factor to obtain power in formal organization, especially for lower participants (Mechanic 1962). Experts maintain power because high-ranking people with formal authority will depend on them for special sills or access to important information. The importance of expertise is magnified in FLOSS projects due to the lack of assigned power. People must prove their special skills or problem-solving abilities, thus obtaining power by others' recognition and trust. The implication of power brought by technical discussion or contribution can be found in both cases. However, it is obviously more manifest in Gaim than Fire. E.g. in Gaim, the 2nd -generation leader was asked once what was the biggest challenge for him, and the answer was "*trying to convince people that I know how to code*." Through a large amount of high-level coding and solution-providing, his expertise was recognized finally by members and users. And this kind of power is embodied in many technical interactions. E.g. many users asked for help with addressing leader's name directly, "*Hi, XXX Could you look at…*" and other developers often stated "*I am not sure. But I will let XXX check it*". Relatively, the expression for the expertise-based privileges is not so much strong in Fire as in Gaim.

4.1.2 Effort and interest

According to Mechanic (1962), there is a positive relationship between the amount of effort a lower-ranking participant is willing to exert in an area and the power he can possess and command. In FLOSS projects, the intensity of effort is an important factor to decide if a joiner can be granted developer's status (von Krogh, Spaeth, & Lakhani, 2003). And the effort exerted is directly related to the degree interest in certain area. In Fire, a member responded to a membership application as following.

Member #F1: *"[…] for becoming a developer, we have sort of an apprenticeship kind of phase where you submit a few patches and one of the developers commits them into the repository. After a few good patches, one of the project leaders (Jason or Eric) makes you a developer and you have commit access. For MSN stuff, the best person to talk to is Mark Rowe (bdash in the irc channel) […]."*

However, the recognition of power driven by effort is affected by the type of activities they involve. It is measured not only based on the number of technical discussions in fora, but more importantly, on the amount of contribution of codes in CVS. Here is a member's self-evaluation who is very active in fora, but aren't involved much in code development.

Member #G4: "*My role is somewhat hard to define. I am not a developer [coder], but I am, in one way or another, involved in nearly every aspect of development. My primary work is not so much to be a bridge between the users and developers, as to be a filter. […] Is my job the most important? I would say it is a useful job, that I am useful to the project, but strictly speaking, I am the least necessary. Gaim is what it is because of the drive and vision of its developers [coder]. […]"*

So enough effort is definitely indispensable for the obtainment of recognition and power in both cases. But it seems Fire assigns more weight to effort than Gaim which is much harsher in expertise.

4.1.3 Role

The role here mainly refers to the role of leader. Usually, there are two ways to become a leader. One comes from the identity of founder, and another kind of leadership must be obtained by the recognition of expertise and effort we mentioned above.

The identity of founder might be viewed as a kind of "formal authority" by some people. However, it is not assigned by some institutions. Usually, the founder possesses strong expertise and put much time and energy in software development, especially at the beginning. But if the founder fails approving his powerfulness, his role will be replaced gradually by certain follower building his power in real work. It is what exactly happened in Gaim whose highly-respected 1st-generation leader took over the role of original founder in no time.

The leader privileges are reflected in multiple ways, such as adding or removing member, deciding successive person, controlling the direction of software development. E.g. in Gaim, the 1st-generation leader often made jokes about his potential successor, e.g. "*Hi, he is lazy now. Do your guys think I should fire him :-) ?*"

However, the privilege is not equal to real recognition of power in FLOSS projects, since people have no responsibility or obligation to accept the command. So only skilled leader can enlarge and exercise his power successfully. This is especially manifest in Gaim due to the high recognition of 1st-generation leader. In a long time, the 2nd-generateion leader stated as following.

<u>*Leader #G2*</u>*: Anyway, this is Rob [*<u>*Leader #G1*</u>*]'s baby. I should let him talk about it. I don't even know he still wants to use this plan in the OS X port.*

4.1.4 Control of commit privileges

Control of commit privileges is obtained by membership, which is the main difference of power between members and non-members. It is a useful tool to ensure the quality of codes and the effectiveness of project management (Reis and Mattos Fortes 2002; Shaikh and Cornford 2003; Lopez, González-Barahona, & Robles, 2004). In FLOSS projects, control criteria not only focus on the feasibility of codes, but also the consistence or similarity of group style and goals. In both cases, we see a lot of patch rejection with such reasons as "*the codes of patch don't follow our style. Please check the FAQ for the coder instruction*" or "*sorry, it is not our focus currently*" and so on.

Thus, the power driven by this of control can easily bring much trouble if not handled well. Anthony (2003) mentioned that it contributes the low rate of volunteer participation in firm-initiated projects. Even in community-initiated projects, some conflicts are generated from the exercise of its power.

4.2 Power Mechanism

In this section, we discuss the ways in which power was exercised in the groups. We examine in particular the way decisions were made, conflicts were managed, problems solved, and the nature of non-work communications.

4.2.1 Decision making

According to Ridgeway (1987), behavior in decision-making teams reflects status and power differentials within the group. There are wide-ranging behavioral differences between people of different power. “Speakers low in power relative to their addressee will tend to use greater amounts of politeness, in comparison to speakers high in relative power. And low power speakers are less likely than power speakers to use positive politeness” (Morand 1996). As well, in traditional organizations or groups, leaders often try to overcome the silence and arbitrariness brought by formal power to ensure equal and universal participation. According to Pavitt (1993), "formal discussion procedures [e.g. reflective thinking; brainstorming as a method of proposal generation etc.] can be a force for democracy in decision-making, and this fact alone may warrant their employment in institutions in which democracy is valued". But such an application of formal procedures is not a common concern in the open source context, because almost everyone has equal nominal position when involved in group activities. As we mentioned before, there is no formal authority or firm hierarchical structure in open source projects, through the title of named leader does bring some privilege to make final decisions based on personal preferences. However, the power can be abused and reduce the morale and participation of community, if it can not be trusted and recognized by others. Losing potential members can be a deathblow for the development of open source project.

In our cases, we find very interesting patterns showing up between two cases based on the behaviors in decision-making. According to Morand (1996), in comparison to members of authoritarian groups, members of more egalitarian group would address each other using a more nearly equivalent level of politeness regardless of formal power differentials.

1) Gaim

Let look at Gaim first. The following example is excerpted from an interaction episode where an outsider from Gnome asked if Gaim wanted to collaborate with them and transferred CVS to certain places.

<u>Outsider #G2</u>: […] So, we need to know how to proceed. The decision is up to you -- are you moving to work more closely with GNOME and want to use the GNOME cvs, or are you more comfortable with Gaim being independently developed as it is now?...Opinions/comments/decisions?[...]

<u>Member #G5</u>: Gaim is not now, and has no intention of becoming a Gnome project. :-) as such, receiving translations here on gaim-devel or in the patch tracker is the ideal way, I or another developer can commit them to cvs […]

<u>Leader #G1</u>: [to <u>Member #G5</u>] Luke -I will handle the response to [<u>Outsider #G2</u>] Christian's message. I'm the maintainer of the project. Let's not forget that.

<u>Leader #G1:</u> [to <u>Outsider #G2</u>] I'm very busy at the moment, but I will give you a response tomorrow. I need to do some thinking about whether or not I want to move Gaim's CVS to another location.

This episode shows obvious power difference in handling different levels of decision. The leader emphasized his position in an authoritative way "*I'm the maintainer of the project. Let's not forget that.*" And he used "*I*" rather than "*we*" to respond to the outsider, which meant he would decide this issue by himself, though it was a big decision related to the whole project. It is revealing that this response generated no objection or challenge from the team. And from the later discussion, this member (*Member #G5)* obviously learned and accepted power formulation when he met similar situation.

Member #G5: I like what I read here. This kind of change is beyond my authority to commit though, you will have to get Sean[Leader #G2] or Rob[Leader #G1] to look at it.

This kind of case not only exists in the interactions between leader and members, but also can be often observed in the interactions between leaders. E.g. the 2nd-Generation leader always dealt with the power difference and relations with previous leader in a careful way. He usually mentioned as following when facing up with high-level decision.

Leader #G2: I doubt anyone would have a problem with being included in Fifth Toe, and of course you have the GPL-given right to freely distribute, but as you want some sort of "official" word, [Leader #G1] Rob's the guy to ask.

In Gaim we also see interesting examples of power exercised between members and users. This excerpt comes from an interaction episode where a user challenged a member's decision to reject a patch.

User #G345: One of the nice things about open source is priorities are set by the users. Just because more important things need to be done has no bearing on this patch - which is already written - being merged.

Member #G5: um... no. Priorities are set by developers. We are volunteers. We are not paid to work on Gaim, we are not paid to please you. We work on Gaim because there are things in Gaim we want, the things we have problems with, the features we want added, the bugs we hit, those are the things that have priority. Anything other features that make it in, any bugs that we fix that we don't experience happen not because of any obligation to users but because we CHOOSE to be nice.

With open source you have the ability to write your own patches, and as importantly, to APPLY YOUR OWN PATCHES. _that_ is the only way in which non-developers have any control over the contents of the programs. Any additional control a project may choose to give users is not an inherent part of open source, but a decision by developers to relinquish control over their own free time, a decision that can at any time be revoked.

Look at the typical description of why things happen in open source: things happen because people have an itch they want scratched, and so they scratch it themselves, they

> *code it themselves. The first step has happened here: someone has provided a patch. But look at the xemacs/emacs situation, the way debian provides qmail, the differences between vi and vim, the differences between nano and pico. You can see it over and over again. People disagree with developers; do the developers give in because the users drive the process? No. the users become developers because developers drive the process.*

From the conversion, we see a clear example of power difference between developers and users in Gaim. The member emphasized the assumptions developers hold and exercised his power to make the patch decision and respond to the user's challenge directly.

2) Fire

Compared to the authoritarian pattern in decision processes, Fire reflects a more democratic or egalitarian style. The issues tend to be posted publicly and developers try to get extensive participation. Usually, it is hard to see any exercise of dominative power in the process, and the members communicate in a comfortable and polite way.

The following example is excerpted from an interaction episode where the nominal leader discussed with other members about a lawyer letter from AOL.

<u>*Leader #F1*</u>*: I just received a letter from AOL's lawyers about trademark and log infringement on Fire […] I am not sure how I can say Fire is AOL Instant Messenger-compatible[…] We need a new name for the buddy list-I suggest Contact List- do others agree?*

<u>*Member #F4*</u>*: […] I think that Contact List sounds OK, but it may sound better as Contacts […]*

<u>*Member #F7*</u>*: […] How about just saying Buddies or Buddies Window […] I guess we could do that […] but at this point, I'm guessing we want to do a release soon, so we should probably stop checking in any new features now. We could release .31.c whatever changes are necessary to appease AOL.*

<u>*Leader #F1*</u>*: Yeah, I agree. Probably by the end of the week […]*

From the conversation, we can see almost no power difference evident. Everyone including leader spoke in a polite and similar way, such as "*do others agree*", "*it may sound* *better as*", "*how about*", "*I guess*" and so on. And we also notice that the leader showed deference and humility, like "*I am not sure how I can…*", "*Yeah, I agree. Probably...*"

Similar attitudes are reflected in handling user requirements in Fire. From the following interaction episode, we can see the equal communication between members and users. There was high consistency and acceptance between them, and no overwhelming power was exercised to make decisions.

<u>User #F121:</u> *[…] Here a question, not necessarily a complaint or an enhancement request. Are there any plans to create change the way the conversation history is stored? [...]*

<u>User #F123</u>: *[…] Some things Id like to have* save to rtf for friends; * always log by name, ignore alias; * or group by alias regardless of service they are chatting on; * search by time constraints […]*

<u>Member #F8:</u> *[…] It is times like this that I wish everyone on this list could program and would submit patches. There are a lot of good ideas here, but the issue is lack of time on part of the developers. See my responses below: […] Submit ideas now, or don't complain when I finish ;) This is a ways off (post 1.0), but I will be tearing apart code soon. You will see the fruits of my last code shred really soon.*

<u>User #F124:</u> *I see three ways that the issue can be resolved […].*

<u>Member #F8</u>: *OK, here is a format that I was thinking about. This is an example irc conversation […] I am thinking about doing the following for savin:[…]*

<u>Member #F8</u>: *OK, as per some suggestions, I have changed the format yet again […]*

The member used a very modest way to communicate with users and tried to satisfy and respond according to users' suggestions and requirements. Many encouraging and thanking words were used, e.g "*there are a lot of good ideas here*", "*submit ideas now, or don't complain when I finish ;-)*" etc.

But we can see some burdens brought to the member. "*It is times like this that I wish everyone on this list could program and could submit patches*", "*ok, as per some suggestions, I have changed the format yet again…*". It is similar as organized anarchy in Decision Process

Models posted by Pfeffer (1981), where no overall organizational goals being maximized through choice and no powerful actors with defined preferences. And the results might end up with inefficiency and overload.

4.2.2 Conflict management

Conflict is a crucial part of cooperative work in organizations. According to Pondy (1967), conflict can be understood in four levels 1) antecedent conditions to some overt struggle (e.g. scarcity of resources); 2) affective states (e.g. tension or hostility); 3) cognitive states (e.g. the perception that some other person or entity acts against one's interests); and 4) conflictful behavior, verbal or non verbal, ranging from passive resistance to active aggression.

Conflict has a close relationship with power distribution. As Katz and Kahn (1978) mention, "more productive of the vertical or hierarchical dimension of organizational life, and this is also the major means for the prevention and adjudication of conflict in conventional bureaucratic organizations". The organizational hierarchy is essentially a gradient of power and authority, concerned with resource allocation, performance commitments setting, task or role assignment and so on.

Conflict is also inevitable in software development, especially in virtual organizations where tasks are loosely assigned, projects are informally managed, and users communicate in distributed places in mainly text-based venues (Elliott and Scacchi, 2003). Although no hierarchy with formal authority preexists in FLOSS projects, power can emerge naturally and generate some hierarchical structure. And conflict can be viewed as a product of this process, e.g. role overlap, rule infringement, power-based interaction style, etc. On the other hand, power can be used as a useful tool to solve conflict, ensuring group cohesion and effectiveness. It also provides good opportunities to enforce group rules/norms, build shared perceptions and help project head to the destined goal.

The situation can be more complex when users obtain much power and are involved intensely in the group activities. As users try to exert their influence on the project, outcomes may be positive or negative. Barki and Hartwick (1994) measured user influence as the amount of perceived influence a user has in systems development. They found a negative relationship exists between influence and experienced conflict, which means individuals with greater influence are more likely to resolve conflict to their satisfaction. So when users have influence, they generally get what they want and experience less conflict. However, if user influence is not recognized by the project or users have different interests or opinions with members, conflict will occur and is more likely to be resolved by the exercise of the stronger power held by members.

In our two cases, different patterns emerge in conflict generation and conflict management.

1) Gaim

Gaim can be viewed as the collision of conflict and power. Due to communication styles and social relations driven by strong and centralized power, the conflicts occur frequently in the interactions. Most of them focus on the task-related issues, e.g. how to develop software, add features, or accept patch. However, the interesting thing is, few conflicts show up between members; most are between members and users.

Let us look at the following conflict episode where a developer tended to reject the patch.

<u>User #G189</u>*: I tested this with various combinations of all the other options. Everything is working well. Please apply this to CVS.*

<u>User #G190</u>*: If the significance of this monumental patch is not clear, let me make it: the existence of the human race depends on it…In short, this is no reason for this box unless the user has enabled the away queue. This patch is sane and is working here [...]*

Form this part, we can see, the users were exercising their power by contributing the patch. And they adopted a confident strategy, e.g. “Everything is working well. Please apply this to CVS,” “this is no reason…”, and “this patch is sane and is working here…”

But the developers had a different opinion about its significance and decided to reject it.

Member #G5: *[…] My reply was along the ideas of John Silvestri, that while this might be a nice option, it is HARDLY as critical as the one post made it out to be. I was pointing out that rather than there being "no reason for this box unless the user has enabled the away queue," that the box is both desirable and necessary for many users for at least the time being […]*

Member #G9: *Oh, no doubt. We all recognize this [...] but right now some other things (completing the conversion to Gtk2, etc.) are taking a higher priority than random UI improvements. The fact that a patch languishes for a little while doesn't mean that it will never be considered, or that the core developers aren't going to put it in without a huge outpouring of grassroots support.*

The power was reflected explicitly by the way of communication when the members explained why they rejected the patch, e.g. “*HARDLY*”, or “*some other things are taking a higher priority than random UI improvement*”.

However, the conflict emerged.

User #G190: *“... I maintain multiple projects - I know how things work. All I meant was the interests of individual users are unique, so when someone goes "I should make this box an option" he could do so regardless of other people's priorities. And then the maintainers can say "hey, sane patch and good idea" and apply it, even if they and others are working on other things…”*

In no way did I mean to nullify the power of the maintainers, merely point out that work can occur tangential to other development and the maintainers can recognize it. I would like to end this thread, now. Please enjoy the holidays.

Member #G9: *“...I agree. I think it's gotten out of proportion, which may be partially my fault. In the meantime, I suggest that those who want this patch apply it manually ... after the Holidays when developers with commit privileges get some free time, maybe it'll go in and that will no longer be necessary. Happy holidays, everyone…”*

So obviously, the patch wasn't got used immediately and the conflict was solved by strong power. From the process, we can see, the conflict is generated by two main reasons 1) the different opinions or interests; 2) the interaction problem driven by power. All people adopted an aggressive way to enforce their own positions and this ended up with a collision of power. E.g. "*In no way did I mean to nullify the power of the maintainers, merely point out that work can occur tangential to other development and the maintainers can recognize it.*"

Though the conflict management is not much successful, from another perspective, we can see the members have very clear plans and preferences about what they are doing and want to do. To some extent, this strategy can ensure the completeness and effectiveness of the whole group under relatively anarchic context.

2) Fire

Compared to Gaim, Fire appears to be a rapport-seeking, polite atmosphere. Almost no true conflicts show up in the interactions. Leader, members and users all choose a positive and comfortable interaction strategy. It tends to avoid many conflicts, or might expresses the conflicts in a silent way like no response. The following example shows how they handled the potential conflict when rejecting a membership application.

User #F90: […] I'm interested in becoming a dev for Fire, and I've already have some code that I would like to contribute. […] My changes are pretty straight forward […] How can I contribute this change to main Fire and/or maybe become a dev on the Fire team...

Member #F9: […] Now it sounds like you did a lot of work here, and I really hate to say this, but it seems like you did a lot of unnecessary work. […] (Specific explanations)[…] Don't get discouraged by what I have said here. We would gladly welcome new developers and there may be something more to what you described that I missed. At the least, make a patch to pt in the patch tracker for the Fire and we will look at it […]

User #F90: Well, clearly I must be doing something wrong. […]

Member #F7: […] Take a look at the developer pages if you are still interested in helping development […]

The developers used a lot of euphemistic words to express their rejection, e.g. "I really hate to say this, but it seems like…" They also persuaded the person by detailed explanations, instead of harsh power enforcement. And power was used implicitly as a positive encouragement, e.g. "*Don't get discouraged by what I have said here. We would gladly welcome new developers and there may be something more to what you described that I missed.*"

4.2.3 Problem-solving

Problem-solving is a kind of basic and routine activity in FLOSS projects. Most of discussions in fora focus on the technical problems and solutions. Through problem-solving, people can show their technical capabilities and obtain power by trust and recognition, and they also can use this power to solve the problem more effectively.

Based on our observations, Gaim and Fire show two kinds of communication style in solving problems. In Gaim, members tend to respond to users in a confident tone, which is especially expressed in the leader with strong expertise and power. Take the following interactions as example:

User #G453: I'm using gaim-CVS at home and like it very much. Now I want to use it on the computers at my university, but they don't have the needed libraries (glib-devel to start, I don't know what else is missing). Is there a chance to build gaim at home in a way - maybe completely statically? - so that I could just copy it? I started using the --with-static-prpls=all flag, but icq seems to be missing then. Furthermore I don't know what to do with all the libraries used.

Member #G11: My instinct says that this will not be possible. Gtk and pango make extensive use of loadable modules for lots of things. I'd suggest installing a little system in your homedir, so build glib, pango, gtk etc with --prefix set to /home/you/sys/ or whatever you want, and then you will get your own /home/you/sys/{lib,bin,share,include} which you can install stuff into, including Gaim and it's library dependencies. If you are short on space build one thing at a time and strip all the resulting .so files. You will need to set maybe CFLAGS to include -I/home/you/sys/include and LD_PATH=/home/you/sys/lib, and maybe one or two others, but it's definitely possible.

Leader #G2: libicq.so is not something you should be using (an old deprecated protocol no longer supported by ICQ) which is why it doesn't compile statically. You should be using the OSCAR protocol, called AIM/ICQ in Gaim. Consult the FAQ. It'd be very *difficult to link Gaim statically with glib, gdk, gtk,etc. You're better off just installing the packages, I'm sure.*

In this conversation, the leader used a dominative way to tell users how to do it, e.g. "Consult the FAQ" rather than "please consult the FAQ". "*libicq.so is not something you should be using*" also expressed implicitly the difference in power and knowledge between two persons. And we also see enough confidence driven by expertise "*you're better off …I'm sure*".

Compared to Gaim, people in Fire are more polite and deferential in solving problems with users and giving solutions as the following example.

User #F231: Both the Synapse and Novel styles are not working in a TOT build (nothing but background is displayed). Anyone else seeing this?

Member #F7: OK, it would really help if you told me which web style you were using. I have fixed Novel and Synapse so they display properly. I have also worked around a crash in Fiat. What style were you using? BTW, the crash in Fiat reported by brett is a WebKit bug, and is Apple's fault, not ours. I have a workaround in place for the moment. I bet yours is the same.

User #F231: Yes, it's in a color-modified Fiat.

Member #F7: OK, Fiat has been fixed. Color-modified doesn't matter; the problem was in the primary CSS file. You should wait for the change to propagate. The first change was a workaround to remove the timestamps from the typing notifications; the second is the real fix and keeps the appearance the same. I have no idea if you will even see the first change or if it will just jump to the second, depend on when anonymous CVS syncs. The issues is WebKit seems to have a crashing issue with certain CSS items (so I guess that means the problem is with WebKit, but I can't exactly fix that, now can I).

4.2.4 Non-work communication

Though more overt power exercised by developers are observed in Gaim than in Fire, it is worth mentioning that this kind of power relation mainly stays in task-related interactions rather than non-work communication.

Contrarily, the working style in Gaim is very casual and intimate. The members and users often joke about each other, and express humor everywhere, even in CVS change log or code notes. For example, the following episode portrays the interactions around a member's birthday.

Leader #G2: *"I'd like to wish Christian (perhaps better known as ChipX86) a very happy birthday."*

Leader #G1: *"I received all of the birthday e-mails in reverse order. How odd. yadhtriB yppaH"*

User #G356: *"Woohoo, go chippy!!!"*

User #G359: *"Dance! [...]everybody Dance!! :D"*

Another episode further shows intimate relationship and strong cohesion between members. What's more, a trust on the leader can be sensed from this conversation.

Leader #G2: *Unfortunately, although Gaim is not associated or affiliated with AOL Time-Warner-I am. And the soonest the Time Warner Cable people can come over and hook me up with some broadband is September 13th-- sometime after 5:00pm. I'll be checking in as much as possible, but won't be too active until then. Just keeping everyone informed*

Member #7: *Noooo! Don't leave us! Guys, did you read that? Sean is retiring :(I'll update the contact info page.*

Member #5: *LOLOL. Let's not go there again. Too many people would believe you.*

Member #7: *Aww, you won't let me have any fun :(Can't we play with the subscribers just a bit? (Kidding of course!)*

Leader #G1: *Sure, it's all fun and games until someone loses an eye.... Then it's just fun you can't see anymore. (According to my crazy grandmother.)*

Leader #G1: *Yeah, enough drama was caused when everyone thought I fired Sean... :-D*

User #564: *Hi! What is internet!? Can I use it to do direct connection!? Thanks! Sean do direct connection plz!! Thanks!*

Compared to Gaim, people in Fire don't make much difference between task-related and non-work issues. In both, they treat each other politely and unassertively. However, this politeness feels like kind of apartness. E.g. people seem not very interested in responding non-work postings. Most of them end with no-response as three following episodes.

The first two original messages were posted by two members of Fire and nobody responded to them.

<u>Member #F3</u>*: Hi all, I'll be gone for a week on vacation...talk to ya when I get back. [No response]*

<u>Member #F6:</u> *I'm going to be gone for the next four days on a ski trip. I should be back around Saturday night. Hope no one needs me by then ... [No response]*

The third one was posted by a leader to welcome two new members and also ended up with voiceless. And this kind of silence almost never happens in Gaim.

<u>Leader #F2</u>*: Welcome to Ken and Eric as the newest Fire developers. They are graduate students from Chicago who will be contributing to Fire as a class project. [No response]*

4.3 Outcomes

Finally, we discuss the effects that the exercise of power had on outcomes for the projects. As noted above. Gaim seems to be more successful than Fire in terms of number of downloads, spin-offs, and in attracting developers. The two projects are very different in terms of language use and the exercise of power. Gaim is more dominative and impolite and Fire is relatively democratic and polite. The question we must ask is how these differences in use of language and power are related to the difference in outcomes.

1. The findings here lead us to speculate that strong, centralized leadership, the assertive exercise of power, and direct language may contribute to effectiveness in FLOSS teams. This seems to contradict much of the conventional wisdom about the egalitarian, democratic characteristics of self organizing, technology mediated distributed teams. However, this idea may

not be as contrary as it seems when we track the nature of power back to its origin. As a self-organizing and volunteer-based context, FLOSS teams are faced with many difficulties, e.g. unstable retention of member, unreliable quality of code, lack of control of working progress etc. All these factors largely affect the success of FLOSS projects negatively. Under these circumstances, the strong and centralized power can benefit effective project management in multiple ways. It can help make final decision quickly, rather than endless discussion. It can solve conflict and balance the interests between different groups. And it can avoid the divergence and break-up of project and push it ahead to the defined goal.

Thus, a democratic, flat group structure may work better for formal organizations, while centralized power might exert a more important effect on improving group effectiveness in self-organizing projects.

2. Strong leadership alone is not sufficient. It must emerge naturally from the voluntary community, based on expertise and competence. It must be action embedded, that is it must be derived from the natural activity of the volunteer group. It cannot be a function of formal appointment. Thus, we would speculate that formal appointment of strong leadership from outside (e.g. corporate sponsorship) would not be effective.

3. Power from expressions of willingness to provide effort is less important than power from expertise demonstrated by actual effort. Willingness to work is not enough for getting recognition, and people must produce high quality code.

4. The power difference between the core development group and the user group is necessary for effectiveness. Though users are important for the development of FLOSS projects, if user influence is too great, it will affect the systematic development progress. And expertise-based power also brings confidence to the community where most of users come to look for technical support. But power difference expressed within the core development group

can be dangerous. It will make developer less motivated to stay. Thus there should be fewer expressions of power difference within the core than between core and periphery.

5. Power distance is necessary, but should not be expressed and reflected in non-work areas. It is harmful to build an intimate and close relationship between people in the community.

5 Conclusion

This paper is very preliminary, and also limited by the selection of cases. Future work will involve more in-depth development of the coding categories identified here in order to evaluate their reliability and validity. As the coding scheme becomes more refined, this will permit both a more quantitative form of content analysis for theory testing to support further conceptual development. It will also provide the tools for a longitudinal analysis that will explore how power emerges naturally and is reflected by language across activities and over time. Finally, a validated coding scheme can form the foundation of an automated analysis of FLOSS transcripts that can alleviate the labor-intensive nature of content analysis. This will make it possible for future work to more easily test the conclusions on a wider sample of FLOSS projects.

REFERENCES

Allen, R.W. & Potter, L.W. 1983. ***Organizational influence processes***. Glenview, IL: Scott, Foresman

Ammeter, A.P., Douglas, C., Gardner, W.L., Hochwarter, W.A., & Ferris, G.R. 2002. Toward a political theory of leadership. ***Leadership Quarterly***, 13 (6): 751-796.

Andrea, B. & Rossi, C. 2003. ***Altruistic individuals, selfish firms? The structure of motivation in Open Source software***. MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/bnaccorsirossimotivationshort.pdf

Anthony, L. 2003. ***How Firm Initiation and Control of Projects Affects Open-Source Development.*** MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/long.pdf

Astley, W. G. & Sachdeva, P. S. 1984.Structural Sources of Intraorganizational Power: A Theoretical Synthesis. ***Academy of Management Review*** 9:1, January 1984, 104-113.

Bitzer, J., Schrettl, W., & Schroder, P.L.H. 2004. ***Intrinsic motivation in open source software development.*** MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/bitzerschrettlschroder.pdf

Bonaccorsi, A. & Rossi, C. 2003. ***Comparing motivations of individual programmers firms to take part in the Open Source movement: from community to business.*** MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/bnaccorsirossimotivationlong.pdf

Bradner, S. 2003. ***Personal communication***, Cambridge, Ma.

Brian, B., Sproull, L., Kiesler, S., & Kraut, R. 2002. ***Community effort in online groups? Who eoes the work and why?*** MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/butler.pdf

Crowston, K., Howison, J., & Annabi, H. 2006 (In press). ***Information systems success in free and open source software development: Theory and measures.*** Software Process: Improvement and Practice Special Issue on Free/Open Source Software Processes. MIT Free/ Open Source Research Community (online). http://floss.syr.edu/publications/crowston2006flossSuccessSPIPpre-print.pdf

Elliott, M. & Scacchi, W. 2003. ***Free software development: A case study of software development in a virtual organizational culture.*** MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/eliottscacchi.pdf

Feagin, J., Orum, A., & Sjoberg, G. Eds.. 1991. ***A case for case study***. Chapel Hill, NC: University of North Carolina Press.

Fiol, C. M., O'Connor, E.J., & Aguinis, H.2001. All for one and one for all? The development and transfer of power across organizational levels. ***The Academy of Management Review***, 262, 224-242.

Hartwick, J. & Barki, H. 1994. Explaining the role of user participation in information system use. **Management Science,** 40 4: 440 - 465

Heckman, R., Li, Q., Xiao, X. 2006. ***How voluntary online learning communities emerge in blended course***. Paper presented at the Hawaii International Conference on System System HICSS-39. Kauai, Hawaii

Higgins, C. A., Judge, T. A., & Ferris, G. R. 2003. Influence tactics and work outcomes: a meta-analysis. ***Journal of Organization Behavior,*** 24 1: 89-106.

Imhorst, C. 2005. ***Anarchy and source code - What does the free software movement have to do with anarchism?*** MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/imhorst.pdf

Howison, J., Conklin, M. S. & Crowston, K. 2005. OSSmole : A collaborative repository for FLOSS research data and analyses. Paper presented at the First International OSS conference, Genova, Italy

Jasperson, J., Carte, T. A., Saunders, C. S., Butler, B. S., Croes, H. J. P., & Zheng, W. J.. 2002. Review: Power and information technology research: A metatriangulation review. ***MIS Quarterly***, 264, 397-459.

Joode, R. W., & Kemp, J. 2001. ***The Strategy Finding Task Within Collaborative Networks, Based on an Exemplary Case of the Linux Community***. Paper presented at Proceedings of the 3rd IFIP Working Conference on Infrastructures for Virtual Enterprises. http://opensource.mit.edu/papers/dejoode.pdf

Kanter, R. M. 1979. Power failure in management circuits. ***Harvard Business Review***, 57, 65-75

Katz, D. & Kahn, R.L. 1978. ***The Social Psychology of Organizations*** (2ed). Wiley & Sons, Inc

Koslowsky, M. & Stashevsky, S. 2005. Organizational values and social power. ***International Journal of Manpower,*** 261, 23-34.

Lakhani, K. R. & Wolf, B. 2003. ***Why hankers do what they do: understanding motivation and effort in Free/Open source software projects***. MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/lakhaniwolf.pdf

Lee, F. & Waguespack, D. 2005. ***Penguins, Camels, and Other Birds of a Feather: Brokerage, Boundary Spanning, and Leadership in Open Innovation Communities***. MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/flemingwaguespack.pdf

López, L., González-Barahona, J.M and Robles, G. 2004. ***Applying Social Network Analysis to the information in CVS repositories***. MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/llopez-sna-short.pdf

Mechanic, D. 1962. Sources of power of lower participants in complex organizations. ***Administrative Science Quarterly***, 7 3, 349-364

Mintzberg, H. 1983. ***Power in and around organizations.*** Englewood Cliffs, NJ: Prentice Hall

Morand, D. A. 1996. Dominance, deference, and egalitarianism in organizational interaction: A sociolinguistic analysis of power and politeness. ***Organization Science,*** 7 5: 544-556.

Morand, D. A. 2000. Language and power: An empirical analysis of linguistic strategies used in superior-subordinate communication. ***Journal of Organizational Behavior***, 213, 235-248.

O'Leary, M. B. and Cummings, J. N. 2002. ***The spatial, temporal, and configurational characteristics of geographic dispersion in work teams***. Working paper, MIT

Pavitt, C. 1993. Does communication matter in social influence during small group discussion? Five positions. ***Communication Studies***, 44, 216-227

Pfeffer, J. 1977. Toward an examination of stratification in organizations. ***Administrative Science Quarterly***, 22: 553-567.

Pfeffer, J. 1981. ***Power in organizations***. Boston: Pitman Publishing, 1-32

Pondy, L. R. 1967. Organizational conflict: concepts and models. ***Administrative Science Quarterly***, 12, 296-320

Raymond, E. S.. 1998. ***Homesteading the Noosphere.*** http://www.firstmonday.dk/issues/issue310/raymond/

Raymond, E. S. 1999 ***The Cathedral and the Bazaar: Musings on Linux and Open Source by an accidental revolutionary***. Sebastapol: CA: O'Reilly & Associates

Reis, C. R.& Mattos Fortes R. P. 2002. ***An overview of the software engineering process and tools in the Mozilla project***. MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/reismozilla.pdf

Ridgeway, C. L. 1987. Nonverbal behaviors, dominance and the basis of status in task groups. ***American Sociological Review***, 52:683-694

Rogers, P. S. & Lee-Wong, S. M. 2003. Reconceptualizing politeness to accommodate dynamic tensions in subordinate-to-superior reporting. ***Journal of Business and Technical Communication***, 17 4: 379-412.

Shafritz, J. M. & Ott, J. S.. 1987. ***Classics of Organization Theory*** (2ed). Pacific Grove: CA: Brook/Cole

Shaikh, M. & Cornford, T.. 2003.***Version management tools: CVS to BK in the Linux Kernel.*** MIT Free/ Open Source Research Community (online). http://opensource.mit.edu/papers/shaikhcornford.pdf

Somech, A. & Zahavy, A. D.. 2002. Relative power and influence strategy: The effects of agent-target organizational power on superiors' choices of influence strategies. ***Journal of Organizational Behavior,*** 232, 167-179.

Von Krogh, G., Spaeth S., & Lakhani K. R.. 2003. Community, joining and specialization in Open Source software innovation: a case study. ***Research Policy***, 32, 1217-1241

Watson-Manheim, M. B., Crowston, K. and Chudoba, K. M.. 2002. ***A new perspective on "virtual": Analyzing discontinuities in the work environment***. Paper presented at Hawaii International Conference on System System HICSS-2002, Kona, Hawai'i, January 2002.

Wayner, P.. 2000. ***Free for all: How Linux and the free software movement undercut the High-Tech Titans***. New York: Harper Business.

TABLE1

Gaim and Fire

Status	**Gaim** (More successful)	**Fire** (Less successful)
Development Status	5 – Production/Stable	5 – Production/Stable
Intended Audience	End Users/Desktop, Telecommunications Industry	Developers, End Users/Desktop
License	GNU General Public License (GPL)	GNU General Public License (GPL)
Operating System	All 32-bit MS Windows, All BSD Platforms, All POSIX Linux, Other	All POSIX, OS X
Programming Language	C	C
Project Member Count	15	12
Project Administrator Count	3	4

TABLE 2

Measurement of Project Success between Gaim and Fire

Criteria		**Gaim**	**Fire**
Use (see Fig. 1)	Average number of download/ per month	137,098	42,336
Activity)	Sourceforge Activity Percentile **(see Fig. 2)**	99.41	94.40
	Average number of posts (tracker: open/close) (see Fig 3)	250 / 222	37/ 34
	Average number of posts (developer mailing list)	97	28
Spin-offs	Number of spin-offs projects	libgaim, and ports to many platforms	not ported
Ability to attract and retain members (see Fig. 4)	Tendency of change in number of developers	Always Increasing membership	Stagnant or falling membership,

FIGURE 1

Gaim and Fire Downloads and Pageviews

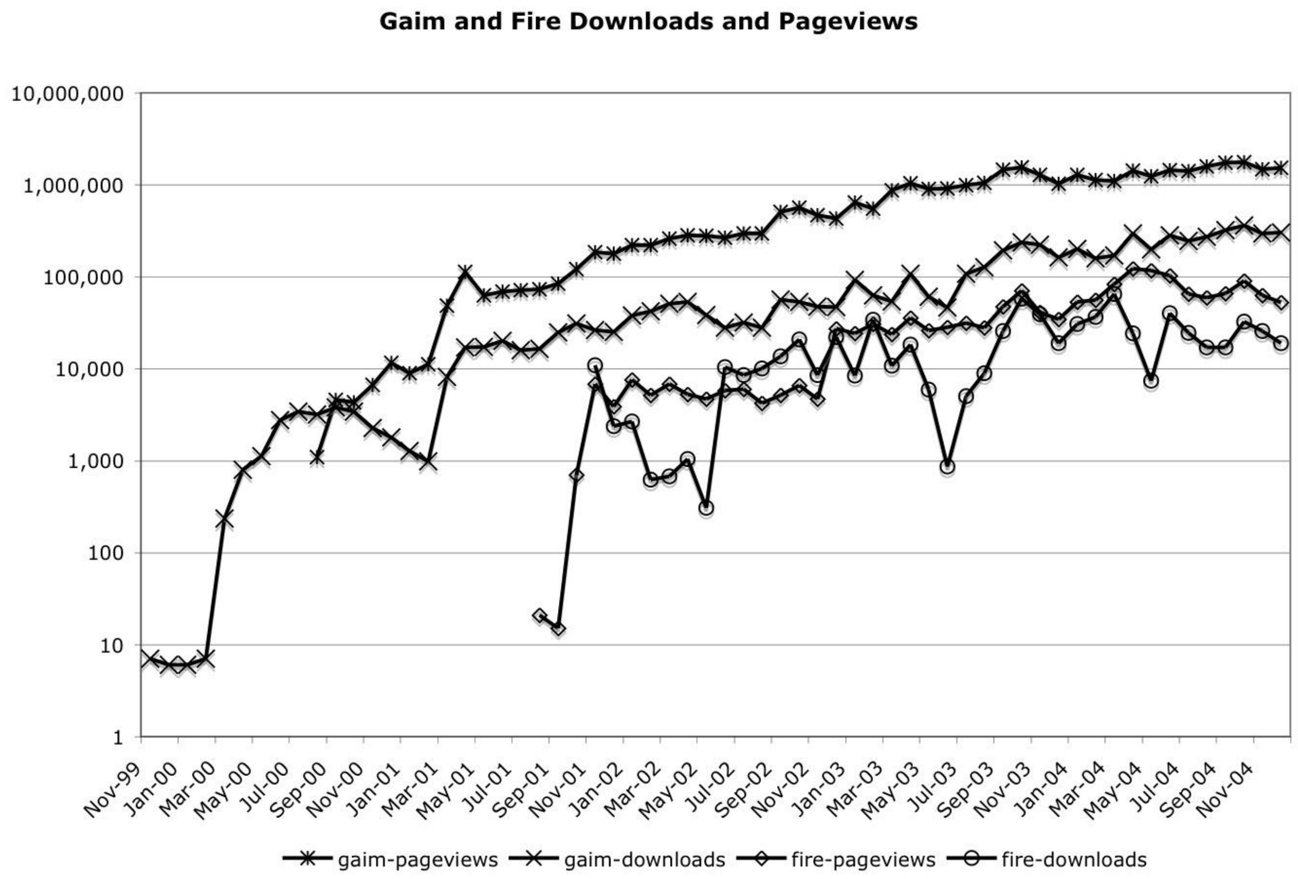


FIGURE 2

Gaim and Fire Message Counts

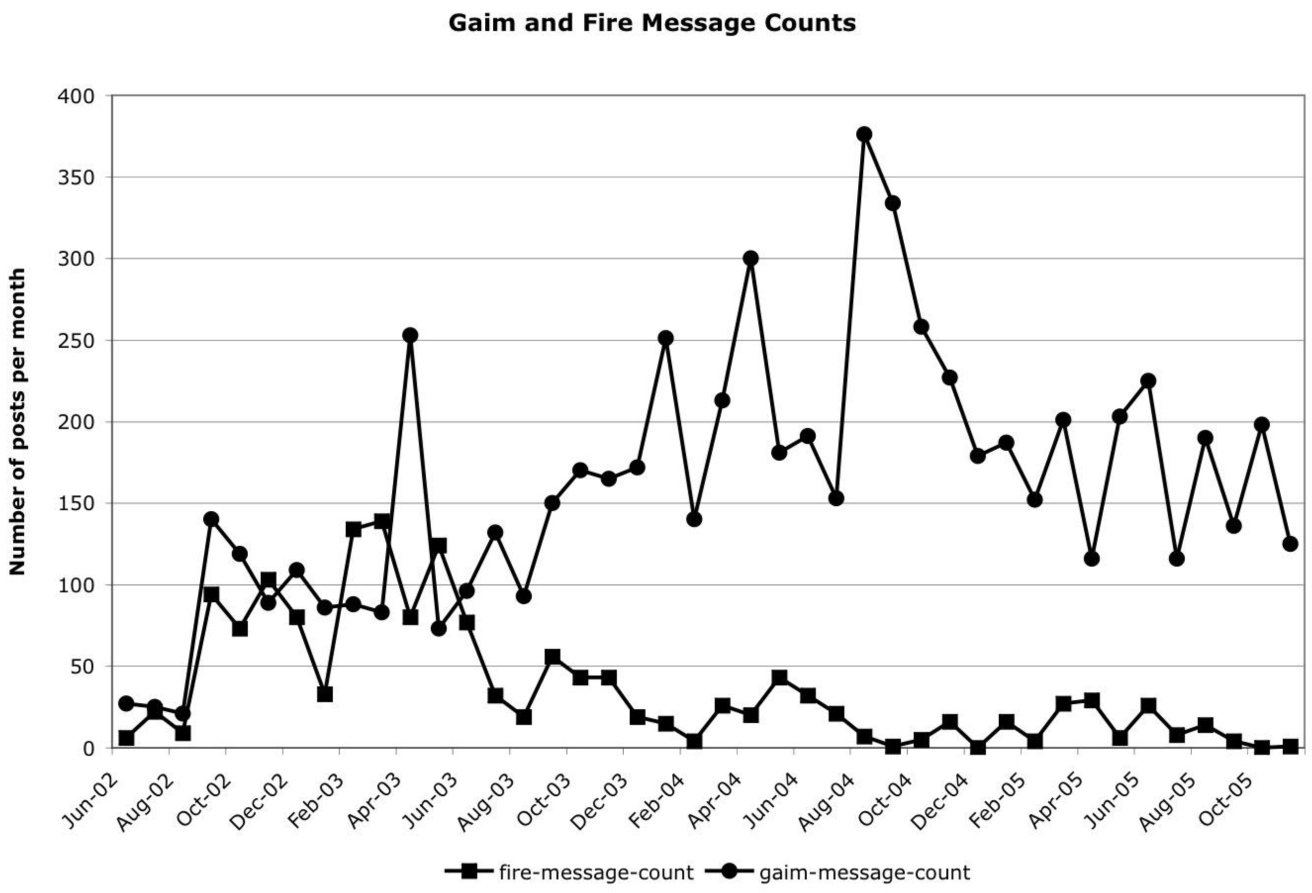


FIGURE 3

Gaim and Fire Activity Percentiles

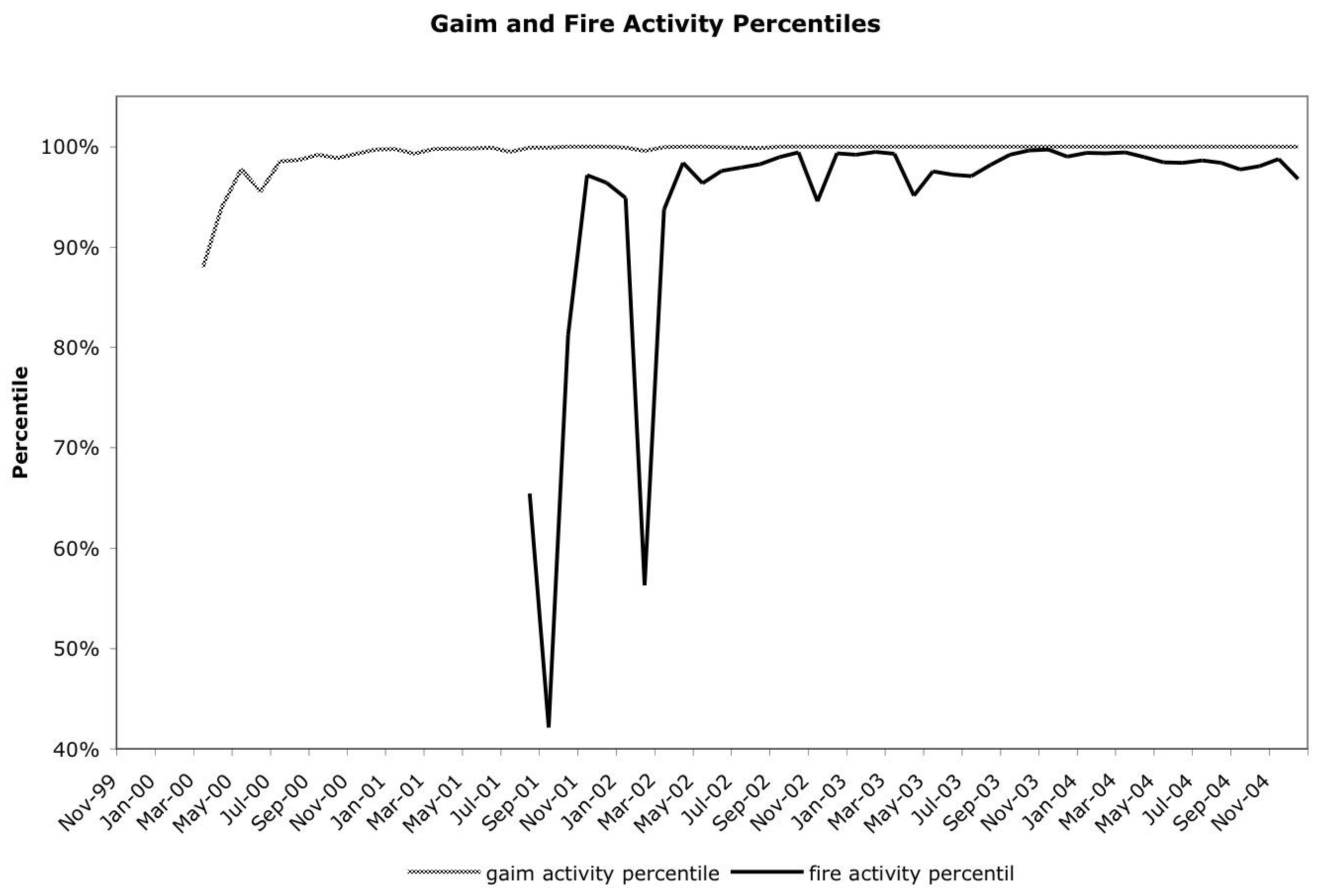
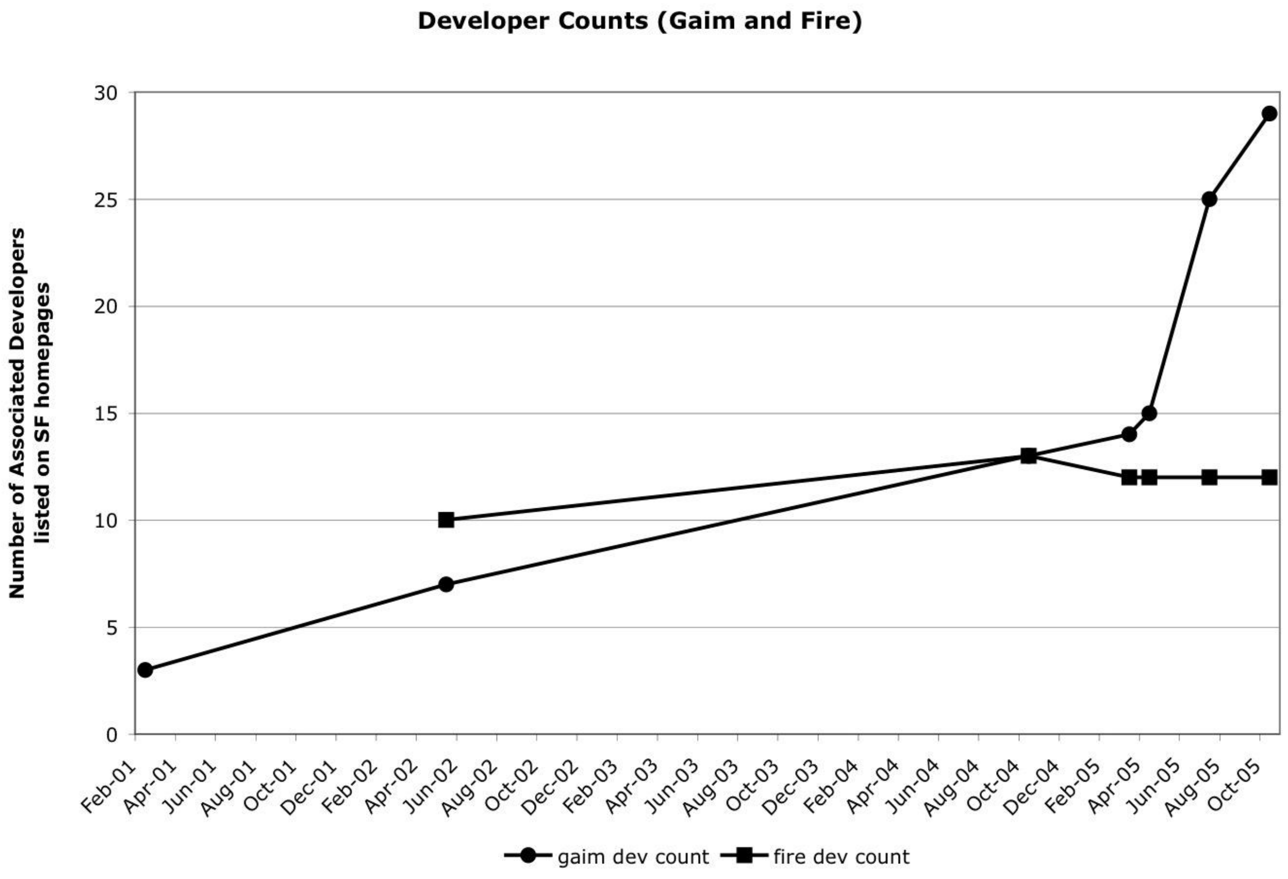


FIGURE 4

Gaim and Fire Developer Counts


Developer Counts (Gaim and Fire)
Number of Associated Developers listed on SF homepages
Feb-01
Apr-01
Jun-01
Aug-01
Oct-01
Dec-01
Feb-02
Apr-02
Jun-02
Aug-02
Oct-02
Dec-02
Feb-03
Apr-03
Jun-03
Aug-03
Oct-03
Dec-03
Feb-04
Apr-04
Jun-04
Aug-04
Oct-04
Dec-04
Feb-05
Apr-05
Jun-05
Aug-05
Oct-05
gaim dev count
fire dev count