Journal of Universal Computer Science, vol. 18, no. 19 (2012), 2625-2645
submitted: 30/3/12, accepted: 6/9/12, appeared: 1/11/12

Division of Effort, Productivity, Quality, and Relationships in FLOSS Virtual Teams: Evidence from the FreeBSD Project

George M. Giaglis
(Department of Management Science and Technology
Athens University of Economics and Business, Greece
giaglis@aueb.gr)

Diomidis Spinellis
(Department of Management Science and Technology
Athens University of Economics and Business, Greece
dds@aueb.gr)

Abstract: Research in virtual teams and distributed work argues that the lack of collocation places an overhead on the performance potential of large, globally distributed teams. In this paper, we revisit this tenet through a case study of Free/Libre Open Source Software (FLOSS) development to demonstrate how globally dispersed FLOSS communities manage to overcome the problem of geographic separation of their members. Our results show that successful FLOSS teams demonstrate a truly global distribution of members, who perform different types of work so as to achieve consistent round-the-clock development, without any apparent ill effects on team productivity and the quality of the resulting outcomes. Cooperation between team members is abundant, especially at more complex work items, and does not seem to be affected by distance; only mentoring relationships appear in some cases to be easier to cultivate between individuals living closer together. These findings challenge the conventional wisdom of research in distributed work, in cases where virtual teams consist of highly skilled and motivated individuals, who leverage the power of communication technologies to overcome problems associated with physical distance.

Keywords: FLOSS, open source development, virtual teams, distributed work, FreeBSD
Categories: D.2.9, K.4.3, K.6.1, K.6.3

1 Introduction

Free/Libre Open Source Software (FLOSS) has arguably revolutionized the way in which software is being produced; the components and processes used for creating and distributing it [Spinellis and Szyperski 04], as well as the collaborative workflow between individuals within or across organizational boundaries for the purpose of software production, have seen a dramatic shift and are challenging the conventional wisdom of the IS community regarding what works and how in information systems development practices. For example, while IS literature has frequently stressed the difficulties associated with distributed software development and remote collaborative work [Herbsleb and Mockus, 03], the counter-examples of successful FLOSS development efforts are too numerous to ignore. The Mozilla web browser, the OpenOffice suite, as well as most components of a GNU/Linux distributions, are

well-known examples of successful systems, each containing millions of lines of code, that have been developed by loosely coordinated teams of software developers who are globally dispersed, without central control or authority, and with practices that differ from those advocated by traditional software engineering and information systems development approaches.

In this paper, we set out to investigate how such globally distributed teams manage to overcome the obstacles associated with geographic distance, time-zone differences, cultural dissimilarity, and coordination overhead to produce software code efficiently and effectively. To do so, we adopt the theoretical viepoint of virtual teams and a case study approach to empirically investigate the historical evolution and current state of one of the largest FLOSS projects in progress: the development of the FreeBSD operating system. We chose the particular setting because of the richness and depth of information available to the researcher, as a result of the large number of developers working together from around the globe and the meticulous process of documenting the process and outcomes of their collaborative effort. These provide a rich setting for extracting systematic data regarding the process and outcomes of distributed collaboration in FLOSS development.

The paper is structured as follows. In the following section, we visit some background research on FLOSS to establish gaps in this emerging literature and place our research in a wider context. Following that, we examine FLOSS communities as virtual teams and set out our hypotheses regarding the expected effects of distance on (a) division of effort, (b) work productivity and quality, and (c) interpersonal relationships within FLOSS communities. Next, we discuss our case setting by outlining the development model, work practices and tools used by the developers that make up the FreeBSD community. We then explain our method of data collection and the results obtained, before revisiting our theoretical propositions to discuss our findings and conclude with insights for further research.

2 Background Research on FLOSS

FLOSS has been the subject of intense interdisciplinary research for more than a decade; Karolak [Karolak, 98] and Carmel [Carmel, 99] wrote the first monographs on the subject, while *IEEE Software* devoted the first journal special issue on open source a couple of years later [Herbsleb and Moitra, 01]. Software engineering researchers were naturally the first to study the new phenomenon, but research from other fields, most notably information systems, quick caught up to reflect the inherently interdisciplinary nature of the new field [von Krogh and Spaeth, 07; von Krogh and von Hippel, 06]. Early research focused, expectedly, on studying the process of organizing remote collaboration teams of software developers with a view to addressing early questions, such as the management of virtual teams [Karolak, 98], technical, organizational, and social work challenges [Lanubile et al., 03], and the emergence of a organic research community around the new phenomenon [Hargreaves et al., 04]. Early researchers were also quick to identify various challenges associated with splitting software development work among various sites [Carmel, 97] and proposed the adoption and use of direct communication channels between developers as a way to alleviate potential problems [Herbsleb and Grinter, 99]. The majority of such early assertions were mostly anecdotal in nature, reflecting

the personal experiences of their authors. More systematic research on the challenges faced by FLOSS teams started to emerge when researchers started using data from free and open source software [Elliott and Scacchi, 03; Sandusky and Gasser, 05], as well as commercial projects [Akmanligil and Palvia, 04; Herbsleb et al., 05], to examine various issues associated with global software development, most notably *factors that affect the success of FLOSS projects* [Subramaniam et al., 09; Hahn et al., 08; Stewart et al., 06].

Another strand of research has focused *on comparing FLOSS development with traditional software engineering methods*. For example, Mockus et al. [Mockus et al., 02] and Reis and Fortes [Reis and Fortes, 02] have examined the processes followed within groups developing the Apache Web Server and the Mozilla web browser, identifying differences in developers' roles, tools used, and activities performed. Scacchi [Scacchi, 02] summarized the differences between these practices and the theoretical tools prescribed in traditional software engineering texts.

Another set of questions that has attracted researchers' attention relates to *the motivation of individuals to participate in FLOSS*, often without pay or any other material benefit, for long periods of time [Subramanyam and Xia, 08]. Amongst the findings of this stream of study, we can highlight the intrinsic motivation of enjoyment [Hertel et al., 03], the altruistic benefits of sharing and learning, either at the individual or the organizational level [Ye and Kishida, 03; Huntley, 03], and the more material benefits of recognition, skill development and better employment prospects [Lerner and Tirole, 2002].

While these studies have shed light into various dimensions of FLOSS, our knowledge regarding the actual effects of the global dispersion of group members on the performance of FLOSS teams and the relationships formed within the community, remains sketchy and anecdotal. Literature has identified factors that could affect these variables in positive ways, such as round-the-clock development [Jalote and Jain, 04], in negative ways, such as lack of face-to-face [Herbsleb and Grinter, 99], and in ways that are indeterminate, such as cultural diversity [MacGregor et al., 05].

Such inconclusive results are perhaps not surprising, given the early stage and exploratory character of such research. However, FLOSS is today mature enough to allow us to delve into these questions more closely. It is interesting to note that research addressing the effects of geographic distance on measures of FLOSS team performance is almost completely missing from existing IS literature. We are aware of only the study of Subramanyam and Xia (2008), who studied drivers of motivation to join a FLOSS community across geographic boundaries – even in this case however, the study was confined to North America, China, and India, and addressed only motivational factors, without looking into the effects of geography on team performance.

Our work builds on previous research and a case study approach to provide additional quantitative data points for the study of global software development and the effects of *geographic dispersion (distance)* between developers on four measures of FLOSS outcomes: *the division of effort* in FLOSS production, the *productivity* of work performed, the *quality* of the outcomes produced, and the *relationships* between team members. In the following section, we theorize on these questions to develop the propositions that are then tested through an inductive case study.

3 FLOSS communities as virtual teams

We adopt a social perspective in order to understand how the effectiveness of FLOSS communities is being affected by the lack of collocation between their members. More specifically, we follow the theoretical streams of virtual teams and distributed work, where issues related to how temporal, spatial, and cultural separation may affect the outcomes of a globally dispersed group of people have been studied in the past.

According to Lipnack and Stamps [Lipnack and Stamps, 00], a virtual team is "a group of people who work independently with a shared purpose across space, time, and organizational boundaries using technology". Thus, the two most prevalent distinguishing characteristics of a virtual team are a lack of collocation and technology-mediated interaction between its members. Virtual teams are frequently used in organizations for a variety of reasons, including the benefits of using the best talent regardless of location and the increased availability of sophisticated technology to support distributed work [Espinosa et al., 03; Paul, 06].

FLOSS development is being performed by groups of people that can be thought of as members of virtual teams, albeit with certain unique characteristics, for example the absence of a solid (non-virtual) background organization, the lack of a predefined and agreed upon set of tasks, and the existence of an open culture where members are free to join or abandon the team at any time. Such characteristics render FLOSS communities more complex than typical intra-organizational virtual teams and call for additional research that will address these specificities. Indeed, researchers have long claimed that the FLOSS movement does not readily lend itself to predictions and explanations of existing theorizations, thus creating a theoretical tension [von Krogh and Spaeth, 07]. Aiming to alleviate this tension, in what follows, we discuss how the lack of collocation between members of FLOSS communities, coupled with the distinguishing characteristics of these virtual teams, are expected to influence measures of team performance (division of effort, productivity, quality, and interpersonal relationships).

3.1 Distance in FLOSS

Virtual team interactions are by definition geographically unrestricted [Johnson et al., 02]. Traditional research in globally distributed teams has shown that such teams have problems maintaining mutual knowledge, which is important for team success [Cramton, 01] – that is why, Olson and Olson [Olson and Olson, 00], after reviewing research on distance collaboration, concluded that 'distance matters'. If distance is an issue in small intra-organizational virtual teams, it is naturally more so as the size of the team and the distance between its members grow. Successful FLOSS teams may consist of hundreds of software developers that reside across five continents and collaborate in the absence of a central authority or a formal management structure. It is therefore important to study how distance may affect the performance, productivity, and interpersonal relationships in FLOSS teams.

On the other hand, it is known that communication technologies can alleviate at least some of the problems associated with distance collaboration [Kiesler and Cummings, 02]. Studying technology-mediated distributed teams, Fuller et al. [Fuller et al., 07] found that group potency and computer collective efficacy are significant

predictors of the efficacy of a virtual team, which is itself predictive of perceptual and objective measures of performance. Similarly, Montoya-Weiss et al. [Montoya-Weiss et al., 01] have shown that temporal coordination, usually facilitated by the advanced use of communication and coordination technologies, can have a positive effect on virtual team performance, while Sarker and Sahay [Sarker and Sahay, 03] have described how communication facilitates the development of the whole virtual team. Hence, the more group efficacy a team demonstrates with communication tools, the less problems it is expected to face in coordinating the remote work of its members.

Previous theorizations have drawn upon observations of various types of organizational virtual teams, which exhibit varying degrees of familiarity with and ability to exploit communication and collaboration technologies to their full potential. Conversely, FLOSS teams, being collectives of expert software developers, are naturally expected to be able to make the most of such tools, thus weakening the effect of distance on group performance. Hence, we expect that the geographic distance between members of a FLOSS team will not have a significant effect in measures of team performance. We will explore the hypo-theses of this statement in the following sections.

3.2 Division of Effort in FLOSS

Arguably, the most fundamental question facing FLOSS researchers is how global development works in practice. We know that it is global teams of developers that cooperate to develop FLOSS, but no research to date has empirically studied the actual distribution of effort across different parts of the world; in other words, whether developers residing in different places contribute different amounts of work or place different emphasis in their contributions to the overall project.

The only other research we are aware of regarding differences exhibited between members of FLOSS teams across regions is by Subramanyam and Xia [Subramanyam and Xia, 08], who studied North American, Chinese, and Indian developers and found significant differences in their motivations to join a FLOSS team. However, research has yet to look into how developers from different regions differ in the amount and type of work they actually perform after they have joined a FLOSS team. Although geographic distance is not expected to be a significant determinant of division of effort in itself, it is known that other factors, for example cultural differences [Hofstede, 01], will affect the behavior of people across world regions. We therefore expect to find some differences in work patterns between regions, but it is beyond the scope of this research to make predictions about what actually drives such hypothesized differences. Instead, our aim is to provide empirical evidence, which will, if realized, will point, along with the findings of Subramanyam and Xia [Subramanyam and Xia, 08], to a need for further research into how cultural factors affect the division of effort within FLOSS teams.

3.3 Productivity of FLOSS teams and Quality of Outcomes

Research in understanding the factors that affect virtual team performance is increasingly important [Hinds and Kiesler, 02]. Information systems research has examined a variety of virtual team performance measures, including decision effectiveness [Schmidt et al., 01], leadership effectiveness [Kayworth and Leidner,

02], and group efficacy [Fuller et al., 07]. Other researchers have looked into how group characteristics, such as trust and communication [Jarvernpaa et al., 04; Sarker and Sahay, 03), can influence group outcomes, while yet others have focused their attention to structural factors, such as temporal constraints [Massey et al., 03] and subgroup politics [Panteli and Davison, 05], and their effects on virtual teams.

In the context of FLOSS, Stewart and Gosain [Stewart and Gosain, 06] have shown how a shared ideology between team members tends to motivate behaviors that enhance cognitive trust and communication quality, which in turn impact the effectiveness of the team positively. The authors point to a few norms and many shared beliefs and values as the binding glue that motivates FLOSS teams and drives their effectiveness. As also pointed out above, we can expect the computer collective efficacy of FLOSS teams to be another important driver of performance [Fuller et al., 07; Montoya-Weiss et al., 01].

We can identify two main measures of performance:

(a) *Productivity*. Globally dispersed virtual teams are able to coordinate and synchronize their activities to enable their project to benefit from 24/7 (*round-the-clock*) development. Moreover, productivity is measured by *development activity*, which is a natural measure of success of a FLOSS community [Stewart et al., 06], as FLOSS projects that fail to generate the necessary amount of activity gradually become abandoned [Hahn et al., 08].

(b) *Quality*. While additional activity is arguably a good proxy of team performance, the quality of this activity is also of paramount importance and needs to be studied if a holistic picture of team performance is to be captured. Quality of code is determined by many elements [Spinellis, 06a] and measuring it is far from trivial [Stamelos et al., 02; Payne, 02]. The main measures used are adherence to code style guidelines and problem reports or bugs found in the code.

Previous research has already hinted that the productivity and quality of distributed software development may not be significantly hindered by the lack of collocation of their members, contrary to the predictions of conventional virtual team theory. For example, in the area of distributed code development (but not in FLOSS), Bird et al. [Bird et al., 09] compared the post-release failures of Windows Vista components that were developed in a distributed fashion against those that were developed by collocated teams and found non-significant differences. This is in line with our thesis that distance will not play an important role in the productivity/quality of FLOSS outcomes. Hence, we expect that *the geographic distance between members of a FLOSS team will not have a significant effect on team productivity or the quality of the outcomes produced*.

3.4 Relationships between FLOSS community members

FLOSS communities are, as all virtual teams, social structures with distinctive properties of organization and emergent relationship building patterns among their members. Research has studied how such self-sustained social networks surface and self-organize out of an initial grassroots, uncoordinated, gathering of individuals. For example, researchers have looked into the processes by which developers find others with shared competencies, values, and beliefs [Crowston and Scozzi, 02; Espinosa et al., 02), how small teams merge into larger ones to reach a self-sustaining critical

mass and the role played by 'linchpin developers' in this process [Madey et al., 04], and how such communities become institutionalized [Sharma et al., 02], thus ultimately transforming the open source software production phenomenon itself [Fitzgerald, 06].

Conventional virtual team research predicts that the formation of interpersonal relationships will be somewhat hindered by the geographic distance between team members. However, in the case of FLOSS, developers have been shown to use other resources (for example, prior collaboration ties – see Hahn et al., 08) to compensate for the lack of physical collocation. Such practices allow members of FLOSS teams to form different types of interpersonal relationships, either *peer-to-peer* (i.e. horizontally, between developers with the same status within the community) or *hierarchical* (i.e. vertically, between expert and novice members of the community) [Spinellis, 06b]. Sociological research has already shown that computer-mediated communication channels are also being used to sustain such ties between individuals engaged in remote cooperative work, thus helping them de-emphasize local organizations and the effects of distance on relationships [Wellman et al., 96]. We therefore contend that *the geographic distance between members of a FLOSS team will not have a significant effect on the establishment of interpersonal relationships between team members*.

4 FLOSS communities as virtual teams

FreeBSD [McKusick and Neville-Neil, 04] is a sophisticated operating system available for a number of modern computer architectures. It is a complete operating system (rather than just a kernel, like Linux) derived from BSD Unix, the branch of Unix developed at the University of California, Berkeley. FreeBSD, known for its stability and reliability, runs the servers of large portals, like Yahoo!, hosting providers, like the Host Department, and embedded applications, like Juniper's routers. Parts of it also form the basis for Apple's Mac OS X.

FreeBSD is the result of open-source software development and maintenance effort performed by more than 600 individuals located throughout the world. The global development effort is coordinated through a number of facilities [Feller and Fitzgerald, 01; Saers, 03]:

- A *configuration management system repository*, based on Subversion and CVS, houses the current version of all the source code and documentation files, maintenance branches of older versions, and more than 15 years of historical data. The complete repository is available for public download and for browsing through a web-based interface.
- A *problem reports database* contains descriptions of open and closed issues, the individuals dealing with them, and details of the resolution history.
- More than 100 open and closed *mailing lists* provide a broadcast mechanism for developers and end-users. The lists cover various development areas (such as security or testing), hardware and processor architectures, and releases of the system.

- A so-called *tinderbox* system continuously performs complete builds of the current source code, providing an early indication of any problems committed to the source repository.
- A *public web site* contains the Developer's Handbook, up-to-date release engineering information, a browsable interface to the version control repository, mailing list archives, and a read-only interface to the problem reports database.
- A *network of machines*, accessible to all FreeBSD developers over the internet, provides developers with a common workspace for compiling and testing their code on different machine architectures.

Developers are mostly unpaid volunteers, although companies with vested interests in the system also have developers contribute as part of their job. An elected *core team* is responsible for deciding the project's overall goals and direction, approving proposals for new developers to join the project, and resolving differences. Separate teams handle release engineering, third-party ports, donations, and security. Developers have the right to modify any part of the system (a so-called *commit privilege*), subject to a few formal and many informal restrictions. For example, developers are not allowed to commit changes while a *code freeze* is in place without prior approval from the release engineering team. Also, heavy modifications on code actively maintained by another developer or changes directly undoing another developer's work are frowned upon.

Developers typically start as enthusiastic contributors of project code; at some point another developer will take interest in their work and recommend them for granting commit privileges. New developers are initially assigned a *mentor* who oversees their work, and approves the changes they make to the code.

We have chosen the particular case setting, as it fits with our objective of drawing upon a theoretical understanding of FLOSS communities as virtual teams and exploring the case-derived empirical data through a theoretically informed lens to provide answers and insights on the questions raised in the previous section.

5 Data Collection, Analysis, and Results

5.1 Method

We derived the version control data from a snapshot of the CVS repository taken on September 14th, 2009, and examined through the CVS client front-end in conjunction with a number of Unix tools we developed and run. Although the development of the FreeBSD source code has moved to use the Subversion system, all changes are mirrored to the CVS repository for the sake of performance and backward compatibility. At the time of analysis, the repository covered about 25 million lines of code and documentation, and contained 1,405,278 commit messages from 594 different committers, made in a period of about 15 years.

We also examined the problem reports database by directly scanning the system files on the project's shell login server, in October 2009. At that time the repository contained around 139,000 reports.

Both CVS commit messages and problem reports are tagged with the login name of the corresponding developer. By establishing a relationship between developers and their locations throughout the globe one should be able to derive a number of

interesting results. To this end, we used a file distributed together with the FreeBSD port of the XEarth application, which contains the latitude, longitude, login name, and location name of many FreeBSD developers. Through two appeals to the developer community, in 2005 [Spinellis, 06b] and 2009, and a wholehearted response, we were able to increase the coverage of the commit lines that could be attributed to a specific global location from 71% on November 17th, 2005 to 83% on September 28th, 2009. We thus secured location data for 394 developers (out of the total 594 ones, a representation of 66.3%). We made no attempt to take into account developers who moved place during the time covered by the CVS data.

5.2 Division of Effort in FreeBSD

The project's 394 developers for which we obtained location data live in 362 different locations in 46 countries throughout the world; the lines of code contributed by each location are depicted as vertical bars on the map in Figure 1. As we can see in the figure, most developers reside in North America and Europe, with pockets appearing in Asia, Australia, South Africa and South America.

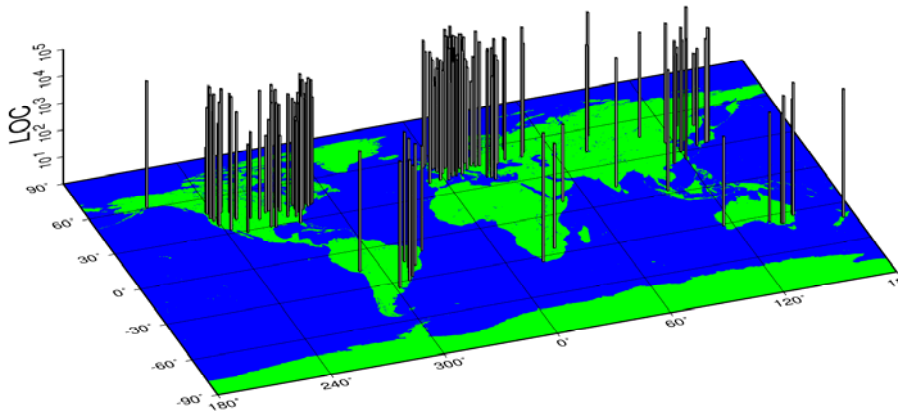

Figure 1: FreeBSD Development effort throughout the world (LOC=Lines of code)

Table 1 illustrates the main countries of residence of FreeBSD developers. USA tops the list, but accounts for only 30% of the global developers' population (44% of the core team membership). The top seven countries make up 62.5% of the population, while the remaining 37.5% is shared by 39 other countries, thus reflecting the truly global scope of the project. The table also illustrates how past and present members of the core team are distributed around the world: they have a dispersion similar to that of the project's other developers. Although the percentage of core team members residing in the US is higher than that of developers, the the core team's US developers reside in eight different states (CA, CO, GA, IA, MA, NY, OR, WA).

Country	Number of Developers (%)	Core Team Members (%)
USA	118 (30%)	11 (44%)
Germany	38 (10%)	1 (4%)
Japan	26 (6.5%)	1 (4%)
UK	17 (4%)	2 (8%)
Canada	16 (4%)	1 (4%)
Russia	16 (4%)	–
France	15 (4%)	–
39 other countries	148 (37.5%)	9 (36%)
TOTAL	**394 (100%)**	**25 (100%)**

Table 1: Countries of FreeBSD Developers' Residence

We can observe two interesting facts in the division of effort across regions. First, the work performed by Asian developers is by an overwhelming proportion (78%) related to the porting and packaging of existing applications. One could attribute this difference to cultural factors, but we don't believe that this work has sufficient data to make such claims. This finding is, however, a strong pointer to the need for further research in this area.

The second interesting aspect concerns differences between maintenance and development work. New development in the FreeBSD occurs in the current, active development branch of the configuration management system. Developers are also encouraged to back-port, where possible, enhancements and error corrections to the stable branches of the older maintained versions following a procedure known as *merge from current (MFC)*. Contributions to the stable branches are more likely to represent maintenance work. Surprisingly, the distribution between the two work types across regions is not the same. Whereas North American work is roughly balanced between contributions to the active and the stable branches (57% versus 43%), the distribution in Europe favors the active branch with about 69% of the lines committed to it. These differences can perhaps be attributed to the fact that there are more FreeBSD-based production systems, such as Yahoo! and Apple's Mac OS X, in North America than in the other regions, where more developers use FreeBSD on their personal workstations. Production systems tend to be based on stable versions, and we can therefore expect North American developers to have an active interest in maintaining them.

The picture is also complicated by the different distribution appearing in the developers who resolve entries in the FreeBSD issue database. Here, Europe, with 45% of the resolved issues, leads North America, which accounts for 30% of the resolved issues, while Asia and Australia trail with 14% and 6%, respectively. Apparently, working through entries of the issues database is a task orthogonal to the one of maintaining the stable versions.

FreeBSD developers are mostly unpaid volunteers, although some have at times been sponsored by various organizations, like Google, DARPA, NLnet, Apple, Isilon Systems, RideCharge, Nokia, and Juniper Networks. However, only a small part of FreeBSD development is visibly sponsored: 0.9% of the commit messages and 2.3% of the corresponding added lines contain a "Sponsored by" acknowledgment. Interestingly, sponsored work (which also includes initiatives like the "Google Summer of Code") trickles down to many developers: 17% of the project's committers have at least one sponsored commit.

All in all, we can say that the FreeBSD project demonstrates a global taskforce of individuals working together from across the world, along with marked differences in the degree of contribution and the types of activity performed across regions.

5.3 Productivity and Quality in FreeBSD

5.3.1 Productivity

As argued earlier, we use two measures of productivity: *round-the-clock development* and *development activity*.

Figure 2 illustrates that the goal of increased productivity through round-the-clock development is indeed realized in FreeBSD. For the past 15 years, FreeBSD developers committed on average 268 lines on every hour of each day; this number fluctuated between a minimum of 164 lines (at 04:00 UTC) and a maximum of 366 lines (at 23:00 UTC), while the distribution of work across time zones does not adhere to a normal distribution typically expected in a non-global setting.

A question related to round-the-clock development is the *granularity* of the work items processed. Is work on a given file passed from one location to the next, are far-away developers cooperating on large modules, or are the development responsibilities divided into different areas? Answering this question allows us to establish a method of round-the-clock development that has worked in practice. To tackle this problem, we observed the commit logs at the level of (a) individual files, (b) complete modules, and (c) the whole system, through various sliding windows covering 8-hour and daily intervals. By looking at commit messages in a given window, we counted the number of days in which commits occurred, the number of days with 8-hour-far commits by the same committers, the number of days with next-day commits, and the number of days with 8-hour-far commits by different committers. These numbers allow us to see both normal work patterns and patterns likely to be associated with round-the-clock development:

- At the level of *files*, round-the-clock development does not appear to be very prevalent. Only in 0.9% of the days was a commit followed after 8 hours by a commit from a different developer; conversely, work periods of the same developer spanning more than 8 hours occur in 1.5% of the days changes are committed into a file. Apparently (and quite reasonably) developers prefer to stretch a long period of work than hand out the work to be completed by somebody else.
- When we turn our attention to *modules*, the situation is reversed: 5% of a module's work days contain commits spanning more than 8 hours by the same developer, while a whole 12.4% of the work days contain commits by different

developers. Here, it looks like developers from different parts of the world cooperate together on the same module, working across different timezones.

- Not surprisingly, the same story appears at the *system* level, where in 97% of the days commits by different developers will span 8-hour periods, while in only 47% of the project's work days will a developer work for more than 8 hours.

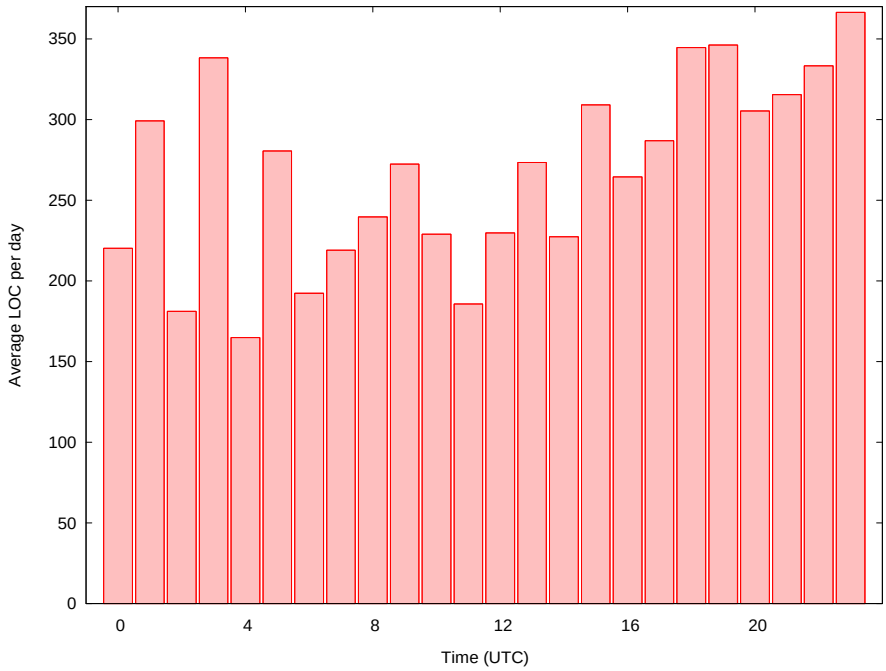

Figure 2: Round-the-clock development

Summarizing, we can claim that in smaller work items, developers tend to work in isolation. However, as the complexity of the work item increases, so does the amount of collaboration between developers from different locations.

As far as *development activity* (our second measure of productivity) is concerned, we examined whether the geographic distance between developers working on a module affects the number of lines that are committed in it. For various modules of the FreeBSD system, we measured the average number of lines committed over a rolling one-month interval, the length of time development went on for a given module, the number of commits performed, the number of different committers, and the average geographic distance between the committers. From the 1,323 modules we examined, we removed modules being developed outside the FreeBSD project (contributed by other groups), and modules with less of 1,000 lines committed over their entire development history. That left us with 504 modules.

As a base case, we examined the correlation between the average number of lines committed per month and the percentage of commits made by different committers. Intuitively one might expect – Brooks's [Brooks, 75] law notwithstanding – that more

people working on a given module would contribute more code. Indeed, we found a positive correlation between those two measures: a Pearson's product-moment correlation of 0.73 in a 95% confidence interval between 0.68 and 0.77. This base case establishes that committers can indeed be used as a proxy for measuring the productivity's input.

Next, we examined the correlation between the average number of lines committed per month and the average geographic distance of the developers committing them. Given that the base case established a correlation between committers and the number of lines committed, we would expect that any relationship between the distance of those committers and the work produced would show up as a correlation in this case as well. However, a two-sided Pearson's product-moment correlation test on those two measures came up only with an extremely low correlation of -0.2 in a 95% confidence interval between -0.28 and -0.11. We can therefore conclude that, in our case, *the geographic distance between developers does not significantly affect productivity*.

5.3.2 Quality

We examined how a large number of geographically dispersed developers might affect the quality of the code produced. If the software's quality deteriorates when software is globally developed, managers should appreciate this problem, and establish procedures for dealing with it. For the purpose of this study, we chose two metrics of quality:

- Adherence to the FreeBSD code style guidelines [FreeBSD, 06], as a proxy for the overall code quality.
- Measurements of the problem reports filed for files committed by globally dispersed developers, as a direct measure of code defects.

As far as adherence to the FreeBSD code style guidelines is concerned, we chose this metric because we could easily measure style adherence by formatting each source code file with the *indent* program configured according to the FreeBSD style guide, and calculate the percentage of lines that *indent* would change (the size of a minimal set of differences between the actual file and the formatted one). Furthermore, by having CVS generate a listing of the source code file with every line annotated with the name of the author who last modified it, we could count the number of developers who had worked on the file.

Armed with these measurements, we used Pearson's product-moment method to examine possible correlations. The correlation coefficient for the 11,040 pairs of measurements was a miserly 0.11 in a 95% confidence interval between 0.10 and 0.13. We therefore see that, in our case, *the involvement of geographically dispersed programmers in the development of code does not affect the quality of the code produced*.

Finally, we examined whether the global development of a file by various developers was associated with an increased number of problem reports filed for it. Such a correlation could indicate that global development in the FreeBSD project leads to an increased number of bugs in the code, due, for example, to communication or coordination problems between the various developers. Although problem reports are kept in a database different from that of the FreeBSD configuration management system, rectified problems are typically marked in a versions control commit message

by a reference to the corresponding problem report (PR). Because serious problem reports are by definition sooner or later rectified, we could establish a measure of the density of problem reports in a file by dividing the number of commit messages tagged with a PR number with the total number of the file's commits. We could then examine the correlation of that ratio with the number of different developers that had committed code to the corresponding file.

We collected data for 41,532 source code files, 630,909 commit messages, and 16,778 PRs. On average, each file was associated with 15.2 commits, 0.40 PRs, and 4.5 different developers. A two sided Pearson's product-moment correlation test between the PR density and the number of committers gave a very small positive correlation between the two values (0.18) in a 95% confidence interval between 0.17 and 0.19. Therefore, data from the FreeBSD project *does not support the hypothesis that global software development is associated with a higher bug density in the code produced*.

5.4 Relationships in FreeBSD

We used data from the FreeBSD project to examine how the distance between developers affects the network of their relationships. We focused on two types of associations between developers: cooperation toward a common goal, and learning in a mentor-mentee relationship. For both types of association, we measured the geographic distance between related developers and compared it to the average distance between developers in the whole community, which we found to be 6,719km[1]. A markedly lower distance between cooperating developers and the average could mean that developers prefer to cooperate with those nearby. From such a result, one could theorize that geographic distance puts a strain in those associations.

We obtained a list of cooperating developers by scanning the commit logs with a rolling window of a single day, looking for different developers who had committed code on the same file within that day. We assumed that such instances would indicate cooperation between those developers, because of the changes' proximity in the code space and time. From the data, we established 12,669 instances of cooperation between developers, and from those instances we kept the 9,606 for which we had at hand the geographic coordinates of both developers. The average distance between cooperating developers is 6,562km, a number very close to the average distance between any two developers. This fact indicates that, in the FreeBSD project, *technical developer cooperation is not influenced by the distance between the developers*.

During the time new FreeBSD developers work with a mentor (typically about a year), they tag all their commit messages with a line indicating the name of the

---

[1] To calculate distances between developers, we used the spherical law of cosines formula, which gives the distance d between two points with latitude δ and longitude λ on a sphere with radius R as,

$$d = R \cos^{-1} [\sin\delta 1 \sin\delta 2 + \cos\delta 1 \cos\delta 2 \cos(\lambda 2 - \lambda 1)]$$

The formula does not take into account the Earth's ellipsoidal shape, but is accurate enough for the purposes of this work.

mentor who reviewed and approved the corresponding change. By reviewing those messages and, also, by using three files where committers are supposed to record their mentor, we established a list of 466 mentor-mentee pairs. From those, we kept the 315 for which we had the locations for both members of the couple, and used the developer coordinates to calculate the distance between the mentor and the mentee (part of the resulting graph is shown in Figure 3).

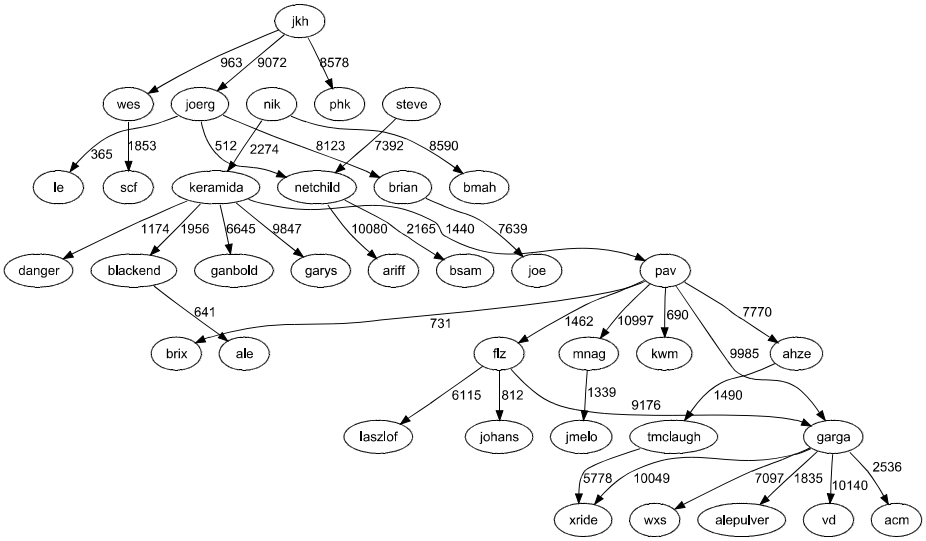

Figure 3: Distances (in km) in a part of the FreeBSD mentor-mentee graph.

As shown in Table 3, which summarizes the km distances between any two developers and developers in a mentor-mentee relationship, the mean and median distances between mentors and mentees are lower than the average, and these differences are statistically significant. However, even in the mentoring case, absolute distances remain relatively high: even the numbers in the first quartile designate distances within a small country or state, not within a city.

	Min	**¼**	**Median**	**Mean**	**¾**	**Max**
All Relationships	0	2,256	7,762	6,719	9,393	19,760
Mentor-Mentee Relationship	0	774	3,401	4,810	8,492	18,650
Difference	N/A	65,7%	56,2%	28,4%	9,6%	5,6%

Table 3: Developer and Mentor-Mentee Distances (in km)

It therefore seems that some mentor-mentee relationships are established between people in the same area (we found 20 pairs living up to 20km apart), but such

relationships can (and do) also work across continents. In other words, *mentoring relationships are somewhat influenced by the distance between participants.*

6 Discussion and Conclusions

The findings in the previous sections indicate that FLOSS software development by a widely dispersed, loosely-coupled team of developers is a practical proposition. The FreeBSD community demonstrates a truly global distribution of members, with different types of work being performed across regions, consistent round-the-clock development, and no apparent ill effects on team productivity and the quality of the resulting code. Ad-hoc cooperation on specific work items is abundant, especially at more complex work items, and does not seem to be affected by distance; however, mentoring relationships appear in some cases to be easier to cultivate between individuals living closer together.

Some of our findings may be counterintuitive and even contradict those of earlier studies that found diminished productivity among distributed teams (for example, Herbsleb and Mockus, 03). As argued in the beginning of the paper, our results reflect the distinctive properties of FLOSS communities, which differentiate them from other forms of virtual teams that have been the basis of earlier theoretical predictions. FLOSS projects are built mainly by volunteers vetted by their peers, hence a number of factors differ from what one would expect to find in an average organizational virtual team: all developers are extremely motivated and highly competent, they freely choose the type and amount of work they will undertake and when they will deliver it, and they are typically also users of the system they help to produce. These factors should be further examined in the context of the relationship between open and closed source software development [Spinellis and Szyperski, 04; Paulson et al., 04].

Our work has demonstrated that, when we examine problems of global software development, there's more to distance than geographic separation. Distance can appear in a number of different orthogonal dimensions. *Physical distance* should be looked in tandem with *cultural distance*, which may be taken to reflect differences in language, social norms and conventions, and predominant work ethic. *Timezone distance* can also crop up in remarkably different ways: developers can share a timezone but be far apart, because they live on different latitudes or because they work on different shifts. Finally, developers' access to various *collaboration technologies*, such as a configuration management system, an issue database, the phone, instant messaging, wikis, and mailing lists, is another underappreciated measure of distance. Such tools become even more important in the case of complex work items, where apparently the need for collaboration is higher (our results showed that developers seem to prefer to work on their own on simpler tasks, where the overhead of online communication is perceived to be higher than the benefit expected). All in all, the findings of this work highlight both the limited explanatory power of existing virtual team theorizations to cater for the FLOSS phenomenon and avenues for extending virtual team and distributed work research to address newly emerging forms of global collaboration, like FLOSS.

The extent to which FLOSS productivity differs from what can be expected in intra-organizational software development, either global or collocated, is another

opportunity for future research. The lack of control groups where developers would work in the same office complex or across the world but under the control and authority of some established organization, does not allow us to make comparisons and verify the extent to which FreeBSD, or FLOSS in general, performance differs from what can be expected in such settings. However, to the extent that many existing commercial software development efforts are also dispersed among physically separated offices or sites, this work has demonstrated that in an environment where developers routinely use a number of essential collaboration technologies, geographic distance can become immaterial – the only exception being mentoring relationships, where communication technologies cannot apparently substitute the benefits of face-to-face communication. Finally, our results indicate that managers should carefully plan the distribution of effort between different areas of the world to address different preferences and skills of developers between regions.

Thus, the results described in the previous sections are relevant to practitioners, and they also open new research questions. Given the generally positive results of this study, commercial software development projects could, at the very least, try to adopt and emulate some of the global development practices of the FreeBSD project. On the research front one could also apply the research methodology of this study to commercial software development projects and see whether the same findings can be replicated there.

The results and insight presented herein can by no means be taken to apply to all FLOSS projects. It is important to recognize that FLOSS is a complex, ongoing phenomenon and the majority of projects to date have either not managed to attract a critical mass of developers to sustain them or have yet failed to release working software [Hahn et al., 08]. However, we have no reason to believe that the situation in other viable, ongoing, successful FLOSS projects will be fundamentally different from the one found in FreeBSD. In order to substantiate this claim though, additional research is needed that will extend the single-case approach we have taken towards more representative (for example, large-scale surveys) or longitudinal (for example, action research or ethnography) paradigms.

Acknowledgements

We wish to thank the members of the FreeBSD community, for allowing us to participate in the project and for providing us with data and comments for this work.

References

[Akmanligil and Palvia, 04] Akmanligil, M. and Palvia, P. C.: Strategies for global information systems development, Information and Management, 42,1, pp.45-59, 2004.

[Bird et al., 09] Bird, C., Nagappan, N., Devanbu, P., Gall, H. and Murphy, B.: Does distributed development affect software quality? An empirical case study of Windows Vista, Communications of the ACM, 52, 8, pp. 85-93, 2009.

[Brooks, 75] Brooks, F.P.: The Mythical Man Month, Addison-Wesley, Reading, MA, 1975.

[Carmel, 97] Carmel, E.: Thirteen assertions for globally dispersed software development research. In the Proceedings of the 30th Hawaii International Conference on System Sciences, p. 445, 1997.

[Carmel, 99] Carmel, E.: *Global Software Teams: Collaborating Across Borders and Time Zones*, Prentice Hall, Upper Saddle River, NJ, 1999.

[Cramton, 01] Cramton, C.D.: The mutual knowledge problem and its consequences for dispersed collaboration, *Organization Science, 12, 3*, pp. 346-371, 2001.

[Crowston and Scozzi, 02] Crowston, K. and Scozzi, B.: Open source software projects as virtual organizations: competency rallying for software development, *IEE Proceedings Software, 149, 1*, pp. 3-17, 2002.

[Elliott and Scacchi, 03] Elliott, M.S. and Scacchi, W.: Free software developers as an occupational community: resolving conflicts and fostering collaboration. In *GROUP '03: Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, pp. 21-30, New York, 2003.

[Espinosa et al., 02] Espinosa, J.A., Kraut, R.E., Slaughter, S.A., Lerch, J.F., Herbsleb, J.D. and Mockus, A.: Shared mental models, familiarity and coordination: a multi-method study of distributed software teams. In the *Proceedings of the 2002 International Conference on Information Systems*, Barcelona, Spain, December, pp. 425-433, 2002.

[Espinosa et al., 03] Espinosa, J., Cummings, J., Wilson, J. and Pearce, B.: Team boundary issues across multiple global firms, *Journal of Management Information Systems, 19, 4*, pp. 157-190, 2003.

[Feller and Fitzgerald, 01] Feller, J. and Fitzgerald, B.: *Understanding Open Source Software Development*, Addison-Wesley, Reading, MA, 2001.

[Fitzgerald, 06] Fitzgerald, B.: The transformation of Open Source Software, *MIS Quarterly, 30, 3*, pp. 587-598, 2006.

[FreeBSD, 06] FreeBSD Project: *Style-Kernel Source File Style Guide*, Dec. 1995. FreeBSD Kernel Developer's Manual: style(9). Available online http://www.freebsd.org/docs.html, 2006.

[Fuller et al., 07] Fuller, M.A., Hardin, A.M. and Davison, R.M.: Efficacy in technology-mediated distributed teams, *Journal of Management Information Systems, 23, 3*, pp. 209-235, 2007.

[Hahn et al., 08] Hahn, J., Moon, J.Y. and Zhang, C.: Emergence of new project teams from open source software developer networks: impact of prior collaboration ties, *Information Systems Research, 19, 3*, pp. 369-391, 2008.

[Hargreaves et al., 04] Hargreaves, E., Damian, D., Lanubile, F. and Chisan, J.: Global software development: Building a research community, *SIGSOFT Software Engineering Notes, 29, 5*, pp. 1-5, 2004.

[Herbsleb and Grinter, 99] Herbsleb, J.D. and Grinter, R.E.: Splitting the organization and integrating the code: Conway's law revisited. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 85-95, Los Alamitos, CA, USA, 1999.

[Herbsleb and Mockus, 03] Herbsleb, J.D. and Mockus, A.: An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering, 29, 6*, pp. 481-494, 2003.

[Herbsleb and Moitra, 01] Herbsleb, J.D. and Moitra, D.: Global software development, *IEEE Software, 18, 2*, pp. 16-20, 2001.

[Herbsleb et al, 05] Herbsleb, J.D., Paulish, D.J. and Bass, M.: Global software development at Siemens: Experience from nine projects. In *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*, pp. 524-533, New York, 2005.

[Hertel et al., 03] Hertel, G., Neidner, S. and Hermann, S.: Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel, *Research Policy, 32, 7*, pp. 1159-1177, 2003.

[Hinds and Kiesler, 02] Hinds, P. and Kiesler, S.: *Distributed Work*, MIT Press, Cambridge, MA, 2002.

[Hofstede, 01] Hofstede, G.: Culture's consequences: comparing values, behaviors, institutions, and organizations across nations, Sage Publications, Thousand Oaks, CA, 2001.

[Huntley, 03] Huntley, C.L: Organizational learning in open-source software projects: an analysis of debugging data, *IEEE Transactions on Engineering Management, 50, 4*, pp. 485-493, 2003.

[Jalote and Jain, 04] Jalote, P. and Jain, G.: Assigning tasks in a 24-hour software development model. In *11th Asia-Pacific Software Engineering Conference*, pp. 309-315, 2004.

[Jarvenpaa et al, 04] Jarvenpaa, S.L., Shaw, T.R. and Staples, D.: Toward contextualizing theories of trust: The role of trust in global virtual teams, *Information Systems Research, 15, 3*, pp. 250-267, 2004.

[Johnson et al., 02] Johnson, S.D., Suriya, C., Yoon, S.W., Berrett, J.V. and LaFleur, J.: Team development and group processes of virtual learning teams, *Computers and Education, 39, 4*, pp. 379-393, 2002.

[Karolak, 98] Karolak, D.W.: *Global Software Development: Managing Virtual Teams and Environments*, Wiley-IEEE CS Press, New York, 1998.

[Kayworth and Leidner, 02] Kayworth, T. and Leidner, D.: Leadership effectiveness in global virtual teams, *Journal of Management Information Systems, 18, 3*, pp. 7-40, 2002.

[Kiesler and Cummings, 02] Kiesler, S. and Cummings, J.N.: What do we know about proximity and distance in work groups? A legacy of research. In Hinds, P. and Kiesler, S. (Eds), *Distributed Work*, MIT Press, Cambridge, MA, 2002.

[von Krogh and von Hippel, 06] von Krogh, G. and von Hippel, E.: The promise of research on open source software, *Management Science, 52, 7*, pp. 975–983, 2006.

[von Krogh and Spaeth, 07] von Krogh, G. and Spaeth, S.: The open source software phenomenon: characteristics that promote research, *Journal of Strategic Information Systems, 16, 3*, pp. 236-253, 2007.

[Lanubile et al., 03] Lanubile, F., Damian, D. and Oppenheimer, H.L.: Global software development: Technical, organizational, and social challenges, *SIGSOFT Software Engineering Notes, 28, 6*, pp. 2, 2003.

[Lerner and Tirole, 02] Lerner, J. and Tirole, J.: Some simple economics of open source, *Journal of Industrial Economics, 50, 2*, pp. 197-234, 2002.

[Lipnack and Stamps, 00] Lipnack, J. and Stamps, J.: *Virtual Teams: People Working Across Boundaries with Technology*, John Wiley and Sons, NY, 2000.

[Madey et al., 04] Madey, G., Freeh, V. and Tynan, R.: Modeling the F/OSS community: a quantitative investigation. In Koch, S. (Ed), *Free/Open Source Software Development*, Idea Group Publishing, Hershey, PA, pp. 203-221, 2004.

[MacGregor et al., 05] MacGregor, E., Hsieh, Y. and Kruchten, P.: Cultural patterns in software process mishaps: incidents in global projects. In *HSSE '05: Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering*, pp. 1-5, New York, 2005.

[Massey et al., 03] Massey, A., Montoya-Weiss, M.M. and Hung, Y.: Because time matters: temporal coordination in global virtual project teams, *Journal of Management Information Systems, 19, 4*, pp. 129-159, 2003.

[McKusick and Neville-Neil, 04] McKusick, M.K. and Neville-Neil, G.V.: *The Design and Implementation of the FreeBSD Operating System*, Addison-Wesley, Reading, MA, 2004.

[Mockus et al., 02] Mockus, A., Fielding, R. and Herbsleb, J.D.: Two case studies of open source software development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology, 11, 3*, pp. 309-346, 2002.

[Montoya-Weiss et al., 01] Montoya-Weiss, M.M., Massey, A.P. and Song, M.: Getting it together: temporal coordination and conflict management in global virtual teams, *Academy of Management Journal, 44, 6*, pp. 1251-1262, 2001.

[Olson and Olson, 00] Olson, G.M. and Olson, J.S.: Distance matters, *Human Computer Interaction, 15, 2*, pp. 139-179, 2000.

[Panteli and Davison, 05] Panteli, N. and Davison, R.: The role of subgroups in the communication patterns of global virtual teams, *IEEE Transactions on Professional Communication, 48, 2*, pp. 191-200, 2005.

[Paul, 06] Paul, D.: Collaborative activities in virtual settings: a knowledge management perspective of telemedicine, *Journal of Management Information Systems, 22, 4*, pp. 143-176, 2006.

[Paulson et al., 04] Paulson, J.W., Succi, G. and Eberlein, A.: An empirical study of open-source and closed-source software products, *IEEE Transactions on Software Engineering, 30, 4*, pp. 246-256, 2004.

[Payne, 02] Payne, C.: On the security of open source software, *Information Systems Journal, 12, 1*, pp. 61-78, 2002.

[Reis and Fortes, 02] Reis, C.R. and Fortes, R.P.M: An overview of the software engineering process and tools in the Mozilla project. In the *Proceedings of the Workshop on Open Source Software Development*, Newcastle, UK, February, 2002.

[Saers, 03] Saers, N.: *A project model for the FreeBSD Project*, PhD thesis, University of Oslo. Available online http://niklas.daers.com/thesis/thesis.html, 2003.

[Sandusky and Gasser, 05] Sandusky, R.J. and Gasser, L.: Negotiation and the coordination of information and activity in distributed software problem management. In *GROUP '05: Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*, pp. 187-196, New York, 2005.

[Sarker and Sahay, 03] Sarker, S. and Sahay, S.: Understanding virtual team development: an interpretive study, *Journal of the Association for Information Systems, 4, 1*, pp. 1-38, 2003.

[Scacchi, 02] Scacchi, W.: Understanding the requirements for developing open source software systems, *IEE Proceedings Software, 149, 1*, pp. 24-39, 2002.

[Schmidt et al., 01] Schmidt, J., Montoya-Weiss, M. and Massey, A.: New product development decision making effectiveness: comparing individuals, face-to-face teams and virtual teams, *Decision Sciences, 32, 4*, pp. 575-600, 2001.

[Sharma et al., 02] Sharma, S., Sugumaran, V. and Rajagopalan, B.: A framework for creating hybrid open-source software communities, *Information Systems Journal, 12, 1*, pp. 7-25, 2002.

[Spinellis, 06a] Spinellis, D.: *Code Quality: The Open Source Perspective*, Addison-Wesley, Boston, MA, 2006a.

[Spinellis, 06b] Spinellis, D.: Global software development in the FreeBSD project. In the *Proceedings of the International Workshop on Global Software Development for the Practitioner*, May, pp. 73-79, 2006b.

[Spinellis and Szyperski, 04] Spinellis, D. and Szyperski, C.: How is open source affecting software development? *IEEE Software, 21, 1*, pp. 28-33, 2004.

[Stamelos et al., 02] Stamelos, I., Angelis, L., Oikonomou, A. and Bleris, G.L.: Code quality analysis in open source software development, *Information Systems Journal, 12, 1*, pp. 43-60, 2002.

[Stewart and Gosain, 06] Stewart, K.J. and Gosain, H.: The Impact of Ideology on Effectiveness in Open Source Software Development Teams, *MIS Quarterly, 30, 2*, pp. 291-314, 2006.

[Stewart et al., 06] Stewart, K.J., Ammeter, A.P. and Maruping, L.M. (2006) Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects, *Information Systems Research, 17, 2*, pp. 126-144.

[Subramaniam et al., 09] Subramaniam, C., Sen, R. and Nelson, M.L.: Determinants of open source software project success: a longitudinal study, *Decision Support Systems, 46, 2*, pp. 576-585, 2009.

[Subramanyam and Xia, 08] Subramanyam, R. and Xia, M.: Free/Libre open source software development in developing and developed countries: a conceptual framework with an exploratory study, *Decision Support Systems, 46, 2*, pp. 173-186, 2008.

[Wellman et al., 96] Wellman, B., Salaff, J., Dimitrova, D., Garton, L., Gulia, M. and Haythornwaite, C.: Computer networks as social networks: collaborative work, telework, and virtual community, *Annual Review of Sociology, 22, 2*, pp. 213-238, 1996.

[Ye and Kishida, 03] Ye, Y. and Kishida, K.: Towards an understanding of the motivation of open source software developers. In the *Proceedings of the 25th International Conference on Software Engineering*, IEEE Computer Society, Portland, OR, May, pp.419-429, 2003.