Who? Where? What? Examining Distributed Development in Two Large Open Source Projects

Christian Bird
Microsoft Research
Redmond, Washington
cbird@microsoft.com

Nachiappan Nagappan
Microsoft Research
Redmond, Washington
nachin@microsoft.com

***Abstract*—To date, a large body of knowledge has been built up around understanding open source software development. However, there is limited research on examining levels of geographic and organizational distribution within open source software projects, despite many studies examining these same aspects in commercial contexts. We set out to fill this gap in OSS knowledge by manually collecting data for two large, mature, successful projects in an effort to assess how distributed they are, both geographically and organizationally. Both FIREFOX and ECLIPSE have been the subject of many studies and are ubiquitous in the areas of software development and internet usage respectively. We identified the top contributors that made 95% of the changes over multiple major releases of FIREFOX and ECLIPSE and determined their geographic locations and organizational affiliations. We examine the distribution in each project's constituent subsystems and report the relationship of pre- and post-release defects with distribution levels.**

I. INTRODUCTION

FIREFOX and ECLIPSE are commonly referred to as archetypes of successful Open Source Software (OSS) development. They are large and complex code bases that have achieved wide scale adoption. ECLIPSE is so successful that it has developed a business ecosystem surrounding it. One obvious question to ask is how such products have come to exist and what types of development processes these projects use. Champions of the open source movement (such as Eric Raymond [1]) claim that open source software defies the constraints of globally distributed development because its main collaboration mechanisms are based on the internet, which is inherently geographically and organizationally distributed.

Open source software systems have been used to study various aspects of software engineering, spanning the spectrum from requirements engineering [2], software quality [3], characterizing the development process [4], investigating code review practices in open source software systems[5], and examining communication practices [6], [7], [8]. However, perhaps because it often seems implicit with the very notion of OSS, few studies exist which examine levels of either geographic or organizational distributed development in open source software. In an attempt to add an additional piece to the OSS knowledge puzzle, we investigate distributed development for two large OSS systems.

In this study, we take a deeper look into the development of FIREFOX and ECLIPSE to study how distributed its development is and what the effects of distributed development are.

We make the following contributions in this paper:

1) We identify and characterize geographic distributed development for two large open source projects at system-wide and also component levels of granularity.
2) We identify which organizations contribute to these projects and report how organizationally distributed these projects are.
3) We assess the impact of geographic and organizationally distributed development using measures introduced in earlier empirical studies of commercial software projects.

With regard to the third contribution listed, a portion of this study represents a replication of an earlier study of distributed development [9] on projects in a different context (OSS vs. Commercial), although the previous study did not examine organizational distribution. In this case, we present a *dependent replication* [10]. As Shull *et al.* indicate, insight into the software engineering question of interest (in our case, the question of the relationship between distributed development and defects) can be gained, whether such a replication has similar or opposite results.

II. PRIOR WORK

In recent years open source software systems have been studied in great detail as they provide a ready repository of real-live software systems, without complications of dealing with software companies for permission to access repositories, legal issues, distribution of results and confidential clauses.

There has been prior work that identifies organizational and geographical attributes of open source projects. We list here the most prominent.

The closest work to ours is that of Spinellis [11]. In this study Spinellis analyzed FreeBSD by investigating the impact of geographical location on code quality. He identified locations of contributions by using the latitude and longitude of developers that distributed with the FreeBSD port of XEarth. In total he was able to attribute 79% of the commit lines to developers with known locations. Based on this data, Spinellis determined that global development allows round-the-clock work, but there are some significant differences between the type of work performed at different locations.

The effects of distributed developers on the quality of code and productivity were negligible.

Tang *et al.* used email metadata such as top level domain and IP addresses in email headers to identify location of contributors to PostgreSQL and GTK+ [12]. They found that in both cases, the majority of contributions came from the United States, followed by Germany. Our method of identification of organization and location does not rely on email, but is also completely manual.

Robles *et al.* looked at how involved companies are in the codebase of projects that comprise the Debian Linux distribution [13]. They used attribution within source files to conclude that 6% to 7% of code in Debian can be attributed to companies, mostly led "by giants like Sun Microsystems, IBM, SAP, Silicon Graphics and AT&T, but also includes more small, focused on libre software companies like Red Hat, Ximian (now owned by Novell) or MySQL." Robles also examined the locations of sourceforge contributors [14] through the time zone and email address information stored at sourceforge. They also found that the top two contributing countries were the United States and Germany.

Interestingly, there are also several specialized conferences for Open source (for instance, OSCon and the International Conference on Open Source Systems) and for distributed development (most notably the International Conference on Global Software Engineering). These conferences are dedicated to these topics individually but we are primarily interested in their intersection.

In contrast to our interest in open source software, many have studied distributed development in the context of commercial software projects.

Cataldo and Herbsleb [15] investigated a large-scale project that implemented 1195 features in a software system. They examined the impact product features, attributes of the feature teams and cross feature interactions on software integration failures. Their results show that factors like the nature of component dependencies and organizational factors such as the geographic dispersion of the feature teams and the role of the feature owners have a complementary impact on software quality. They also found that cross-feature interactions, quantified by the number of architectural dependencies between two product features, were a major driver of integration failures.

Ramasubbu *et al.* [16] studied 362 projects from four different firms to assess the impact of impact of project-level configurational choices of globally distributed software teams on project productivity, quality, and profits. They identified that imbalances in the expertise and personnel distribution of project teams significantly helps increase profit margins but a profit oriented imbalance could also significantly affect productivity and/or quality outcomes. They provide recommendations and insights for managers and companies to make the correct choices to help enable successful projects.

Cataldo and Sangeeth [17] examined the impact of process maturity and geographic distribution on software quality in a multi national software development organization and and found that there was indeed a large impact. Further they found that as work becomes more distributed the benefits of process maturity diminish.

Prior work by Bird *et al.* [9] investigated the distributed development of Windows Vista across different sites at Microsoft. The study showed that distributed development does not affect software quality when organizational and process changes are put in place to enable software development.

III. Research Questions

Unlike commercial software in which contributions come from employees of the same company or in some cases from contracted entities, OSS projects accept work from nearly any organization (provided that such work is of sufficient quality and is legally acceptable). Research at Microsoft [18] and in other commercial settings [19] have found that when a component or work item is spread organizationally, quality and productivity suffer. While OSS projects do not typically adhere to an organizational structure as strictly as in the commercial world, we *can* still identify the organizations that are contributing to the project.

Research Question 1: What organizations contribute code and how organizationally distributed are the projects?

In addition, we are also interested in understanding and characterizing the level of geographic distribution of OSS projects. This is also required in order to examine the relationship of distribution to quality. We therefore ask:

Research Question 2: How geographically distributed are the software projects?

Lastly, once we have understood and characterized the level of geographic and organizational distribution in OSS projects, what are the effects, if any, on software quality? Are the results commensurate with prior studies, indicating a more general phenomenon that is not development process specific or do the effects of distribution on quality differ in OSS?

Research Question 3: What is the effect of geographically and organizationally distributed development on software quality?

Our hope is that by examining OSS projects in a way similar to previous studies in other contexts, we can gain a deeper understanding of the effects of distributed development (e.g., delay or quality) and how these effects change with domain and process.

IV. Data Collection and Analysis

In this section we discuss the collection of data as well as the techniques used to analyze the data. As the novel data that we collected includes the organizational and geographic information for developers in Eclipse and Firefox, we have made this data available on the PROMISE Software Engineering Repository [20][1].

A. Data Collection

For our analysis, we collected a number of types of data. We gathered data from source code repositories and bug databases and also determined the locations of developers.

As an OSS project, Mozilla maintains a publicly available source code repository, which contains the sources for the Firefox browser. We mined the changes from this CVS repository, which comprised a total of 1,147,175 changes from March, 1998 to March, 2008. In our analysis, we only examined changes to the files that were included in two major releases of Firefox, 1.5 and 2.0 during their respective development cycles. There were 6,151 and 6,211 C and C++ source files that shipped as part of Firefox versions 1.5 and 2.0 respectively and a total of 44,877 changes made to these files during development of these releases. We also mined the changes from the Eclipse CVS repository, which comprised a total of 920,989 changes from April, 2001 to February, 2008.

We found that 227 accounts made commits to the Firefox codebase. However, the 77 most active contributors accounted for 95% of the source code commits. None of the other contributors accounted for more than 0.2% of the commits to Firefox during these development cycles. Although there were 208 Eclipse CVS accounts, we found that some people had multiple accounts. After dealing with the account aliasing, we found that 190 unique people made commits to main project repository. Similar to Firefox, Eclipse contributions were skewed such that the 100 most active contributors accounted for over 95% of the commits and non of the remaining 90 contributors made more than 0.12% of the total commits.

We use Firefox pre and post-release defects that had been mined previously for each of the C/C++ files as a part of this study. For our Eclipse bug data, we mined the project's bugzilla repository. This database contains information such as severity, version of the product, priority, assignments, resolution information and timestamps for 221,518 bug entries. We only include entries that are marked as bugs (e.g., no feature requests) and that are assigned to the Eclipse main platform.

We used automated techniques to link closed bugs in the database to the commits that fixed them in the software repository [21]. Each bug in the database includes the date that it was opened and also the version of Eclipse that the bug occurred in. We use this information along with the release dates of each version of Eclipse to categorize bugs into pre-release and post-release for each of the versions of Eclipse released during the period of study. Eclipse has a rigorous policy of manually attributing bug IDs in their log messages. We manually used text similarity and information from the internet to link CVS accounts with bug database accounts in Eclipse using techniques introduced by Bird *et al.* [22].

[1]http://promisedata.org

B. Locating Developers

We are interested in where the developers are both in terms of organization (who do they work for) and geography (where do they work). We note that just because a developer works for a particular organization *does not* mean that the organization itself is a formal contributor. If a student from MIT or an employee from BEA commits code, that in no way indicates sponsorship from their organizations. However, we expect that ease of communication, commonality of goals, and cohesiveness of changes will differ based on whether developers are from different organizations or not. This information is not generally made public anywhere so deeper investigation is required. Fortunately, for each developer that contributed directory to the source code repository, we can extract their email address, name, and history of contributions. We identified the top contributors that made 95% of the commits to the each project's source repository.

We used a number of techniques to determine the geographic and organizational information for the developers.
Email domain names — Many email addresses contain geographical or organizational information in the domain name. For instance, `bzbarsky@mit.edu` indicates a contributor from MIT. Companies like IBM are apparent from the domain name. Often the country is also indicated in the domain (for instance `enndeakin@sympatico.ca` is in Canada). Although the granularity is coarse (*e.g.* only narrowing down to a country), these give some indication of where else to look for identifying information.
Social Networking sites — LinkedIn is a common professional networking site. Since most primary developers are professionals, many of them were on LinkedIn. Their profiles indicate the metropolitan area that they are living in and often list job history so we can see where they were at the time of commits. In cases where a name is common, such as "Jeff Brown", it is possible to filter on employer or job domain. In addition, other networking sites such as Facebook, which often indicate geographical location or employer, were also helpful.
Blogs — Since developers are heavily technical and active in Eclipse or Firefox community, many maintain either technical programming or personal blogs. It is usually apparent where they live and who they work for from their postings on these blogs. As an example taken from our investigations, "I got back to MIT to begin the Spring Semester last week..." indicated an MIT student that made contributions.
Emails — Often information is contained in the body of an email about the location of the sender. People may refer to others and include their location. For instance "...as Debbie (Ottowa/Can/IBM) has stated previously, the UI can't..."

indicates that Debbie (whose full name we identify earlier in the email thread) works at IBM's Ottawa site.

Presentations and Conferences — Many developers attend conferences or give presentations to others. The title slide in presentations online often included the name of the presenters along with their location of employment. Online conference programs also usually give a brief bio of the presenter which included their location and employer.

Direct Communication — In rare cases, we contacted contributors directly. As an example, when searching for Billy Biggs, a developer for ECLIPSE, we encountered a web site for someone with that name that listed ECLIPSE as one of his projects. We contacted him through his site and asked questions regarding where he lived and who he worked for (as well as the same information for other hard to locate contributors) [23].

Company sites — Some companies contain biographical pages for their employees or allow employees to create their own pages (e.g., Microsoft Research has this facility for its employees). Often there is a link to the site that the employee works at and the location can be inferred from that.

Web Articles — In one case a web "journalist" for an online news site had written an article about ECLIPSE and mentioned that he'd talked to an ECLIPSE developer where he worked in San Francisco at BEA. This indicated both the location and employer of the developer.

Association — For some individuals, we could not find their location directly, but another person whose location we had already determined indicated that they worked at the same location, either in a blog entry or an email. Thus we were able to determine the location by association.

SCM History — File logs from the source code management system can also provide valuable clues for identifying contributors. In one notable case for ECLIPSE, the CVS user name for an unknown contributor was `torres`. Unfortunately, we were able to identify four people with the last name of Torres that are associated with ECLIPSE and were unable determine which person used the `torres` CVS account. Upon examining the CVS logs, we found that the vast majority of the files that `torres` committed to had to do with SWT (The GUI toolkit in ECLIPSE). This additional information allowed us to improve our search and we were immediately able to determine which of the four corresponded to the committer (in this case Elias Torres) along with his blog which indicated his location.

For a number of sites, we were able to get the exact address of the company. In cases where we were unable to determine the organization that a developer was affiliated with (or when the developer indicated that he was self-employed or not employed) we had to settle for the city. For each person we identified the city, state or province (where such existed in the country), country, and company. From these, we determined the latitude and longitude, used for time zone data and distances between sites.

Using these methods, we identified the locations for the top 77 contributors in FIREFOX and the top 100 contributors to ECLIPSE. In both cases, the contributors account for over 95% of the contributions to the project.

C. Measures of Distribution

We include a number of measures of geographic and organizational distribution.

Distribution Level

Similar to a prior study on distributed development in a commercial setting [9], we categorized the level (which we denote LEVEL) of geographic distribution for each component based on the smallest geographic entity that we could trace 75% of the commits back to. Thus, if we could identify one city from which 75% of the commits came from, we categorized the component as distributed at the city level. If not, we examined the different nations that contributed to the binary and then the number of continents that contributed. If we were unable to identify a continent that contributed at least 75% of the commits, we categorized the component at the worldwide level.

The threshold of 75% was chosen for two reasons. First, prior studies [9], [18] used this threshold and using the same value allows for an equitable comparison between our findings and previous findings. Second, our measure is discrete rather than continous and using a threshold of 100% would allow just one commit to a component make LEVEL jump from being distributed within a city to being distributed worldwide. We did not want LEVEL to be susceptible to such phenomena.

Spatial and Temporal Dispersion

Following the methodology of Cataldo and Nambiar [17], we use the definition of *Spatial Dispersion* introduced originally by O'leary and Cummings [24]. In the dispersion equations, N is the total number of developers that contributed to a component, L is the set of locations of the developers making contributions, and N_i and N_j are the number of developers at locations $i, j \in L$ respectively. For our measure, SPATIAL, KM_{ij} is the distance in kilometers between location i and location j.

$$\textsc{Spatial} = \frac{\sum_{i,j \in L} \mathrm{KM}_{ij} \cdot N_i \cdot N_j}{\frac{N^2 - N}{2}} \quad (1)$$

Similarly, in the definition of Temporal Dispersion, TEMPORAL, TZ_{ij} is the absolute value of the timezone difference between locations i and j in hours.

$$\textsc{Temporal} = \frac{\sum_{i,j \in L} \mathrm{TZ}_{ij} \cdot N_i \cdot N_j}{\frac{N^2 - N}{2}} \quad (2)$$

For both definitions of dispersion, if developers in only one location made contributions to a component, the dispersion is zero. Similar to Cataldo and Nambiar, we measured distances between sites in the same city (when such resolution was possible), but did not measure distances between buildings at the same site.

Organizational Distribution

We define two organizational measures that draw on the results of Nagappan *et al.* [18].

First, we identified the number of organizations that contributed to a component as ORGANIZATIONS. Different organizations may have different agendas in mind when making contributions to an OSS project. Further, coordination between developers in separate organizations is most likely more difficult than coordination within an organization due to shared culture, background, and in some cases, geographical colocation. We therefore expect that components that are contributed to by multiple organizations will exhibit higher defect rates.

Second, ORGANIZATIONAL OWNERSHIP measures the proportion of commits to a component contributed by the "owning" organization (the organization that made the most commits). If one organization has a high level of ownership, we expect that the organization has a vested interest in its success. Further, the organization may act as a point of contact for the component. We hypothesize that this will lead to fewer defects.

D. Quality Analysis

In order to study the relationship of each of these measures with failures, we use them as independent variables in a linear regression model using pre and post-release bugs as dependent variables. In addition, we also include control variables in our models so as to control for the effects of code metrics known to have a strong relationship with failures [25]. We measure code size using lines of source code which we refer to as LOC. CHURN is the sum of the number of source lines added and deleted (a modified line is counted as one line removed and one line added) in a component between releases. Finally, COMPLEXITY, measures the cyclomatic complexity of the code. LEVEL is included in our model as a series of binary variables, as it standard for categorical data [26]. We include NATION, CONTINENT, and WORLDWIDE, only one of which can have value of 1 for any component. Components developed in one site have all of these variables set to 0, as same site development is the baseline and included in the intercept.

Unfortunately, including all of the control variables, independent variables (our distribution measures), and dependent variables (pre and post-release defects) leads to potential problems. First, the validity of linear regression results rests on key assumptions holding. One of these assumptions is that the residuals in the models are normally distributed [26]. In all cases, we found that pre and post-release defects suffered from skew and the residuals were not normally distributed. In addition, some of our independent and control variables also were skewed and caused similar problems. To mitigate this, we performed a logarithm transformation on these variables before using them in the regression models. We indicate where this transformation was used in our discussion of results in the following sections.

Second, we note that some independent or control variables may be highly correlated. Models using these variables will potentially overfit on the data, leading to a problem known as multicollinearity [26]. We deal with this problem by examining the Variance Inflation Factor (VIF) for our linear models and removing variables with VIF values at 5 or above. As an example, in our models, we found that COMPLEXITY was highly correlated with LOC in all models (a not surprising result, given prior studies that also show a relation between LOC and many code metrics [27]). We therefore choose the best model that considers only one and use LOC.

We hasten to point out that although regression has been used with defect data to build accurate defect prediction models (see, for example, extensive work by Weyuker and Ostrand including [28]) our purpose is *not* to use regression for defect prediction. Rather we use it to *identify the relationship*, if any, between our measures of geographic and organizational distribution and defects when controlling for known factors.

V. FIREFOX RESULTS

We answer the three previously stated questions regarding distributed development in FIREFOX in this section.

A. Organizational Distribution

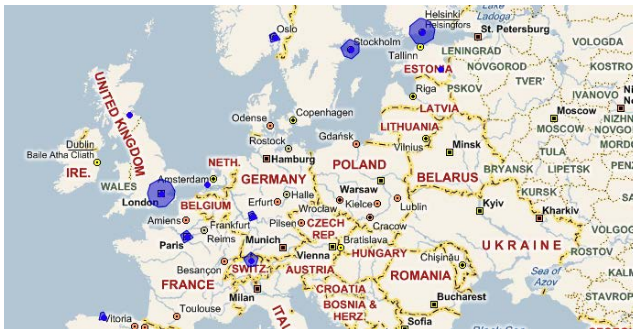
We are first interested in how distributed FIREFOX is along organizational boundaries. We used the organizational affiliations of developers to determine what proportion of FIREFOX development (measured by number of commits) come from different organizations. Note that the organization itself is not necessarily the contributor. For instance, University of Queensland and MIT indicate contributions made by students of those respective universities. The category labeled "self" represents people who identified themselves as self-employed contractors or "Full Time Hackers" that were not affiliated with any company. In addition, there were a number of individuals that we were able to identify geographic information for, but could not determine organizational information about. These make up the "Unknown" category and are not included in our organizational analysis.

Figure 2 shows the breakdown of contributions for release 2.0. We do not show breakdowns for release 1.5 because there difference was minimal. Our data shows that Mozilla Corp is the largest contributor to FIREFOX followed by Google. However, there are a number of contributions made by other companies with an interest in the browser market as well as academic institutions.

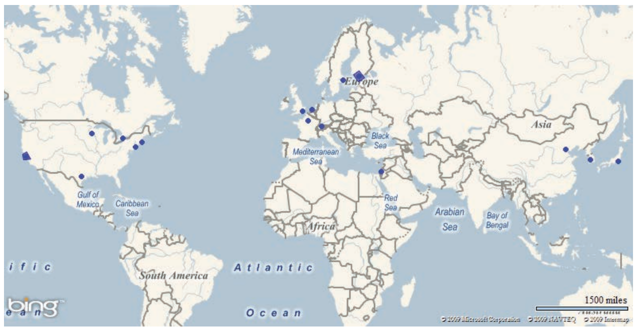
Therefore, our answer to **RQ1** is that *while Mozilla Corporation is by far the largest contributor to* FIREFOX*, a diverse group of companies and organizations account for more than half of the source code contributions.*

B. Geographic Distribution

We also examined how FIREFOX is geographically distributed. We were not able to determine building information for the developers in FIREFOX. Therefore we categorize the geographic distribution of components into four levels: same site, same nation, same continent, and worldwide.



(a) Contributions to FIREFOX from locations in Europe. Blue circles indicate locations contributing to FIREFOX. The number of contributions is proportional to the area of each circle.



(b) Contributions to content event handling in FIREFOX from locations across the world. Blue circles indicate locations contributing to FIREFOX.

Figure 1: Maps of FIREFOX development

FIREFOX 2.0 Contributions

Organization	Commits	Percentage
Mozilla Corporation	9551	47.6%
Google	2102	10.4%
MIT	1716	8.5%
Nokia	758	3.8%
Intel	705	3.5%
Netscape	631	3.1%
IBM	334	1.7%
XForms	277	1.4%
Sun	209	1.0%
Korean Adv. Inst. of Tech.	148	0.8%
Macalester Univ.	104	0.5%
Univ. of Queensland	94	0.5%
8×8	79	0.4%
Thales	72	0.4%
University of Tartu	54	0.3%
Red Hat	8	0.0%
Self	710	3.5%
Unknown	1525	7.6%

Figure 2: Contributions to FIREFOX 2.0 by organization. "Self" and "Unknown" are listed at the bottom because each individual in these categories comprise his or her own organization. Total sums to 95% because we did not determine information for the 100+ developers contributing the remaining 5%.

As Figure 4 shows, FIREFOX is quite geographically distributed. Unsurprisingly, California is the geographic region that makes the largest number of contributions. This is due to the fact that Mozilla Corp, Google, Sun, Netscape/AOL and Redhat all have offices and in some cases, headquarters, in the the California Bay Area, near San Francisco. Despite this, we see that contributions are made from a number of countries throughout the globe, with a focus in North America and Europe. Note that each location depicted may include multiple sites. For instance, there are two unique sites in Ontario, Canada, and three in Israel.

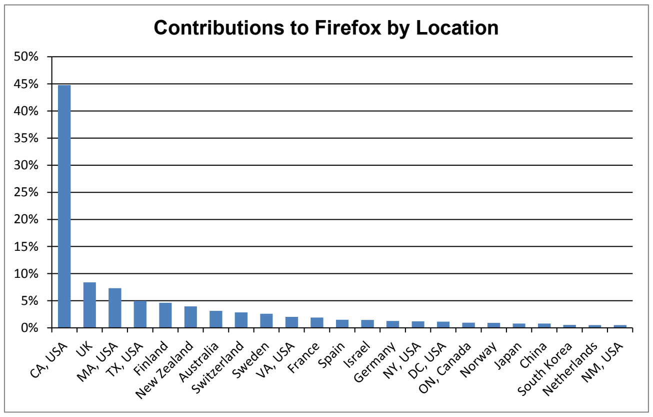


Figure 4: Contributions to FIREFOX by location.

While almost half of the contributions for FIREFOX *come from the California Bay Area, the other half is distributed worldwide, with a focus on Europe and North America.*

We also visualized contributions geographically. For example, Figure 1a shows a map of Europe with FIREFOX development sites indicated as blue circles. The area of each circle is proportional to the number of commits to FIREFOX from that location.

When broken down by component, the level of geographic distribution is varied. For instance, the map in Figure 1b shows development for the code that handles events within displayed internet content. The development of this module was split across three continents and 16 different countries/states. At the other extreme is the inter-process communication code, which was developed almost exclusively within Mozilla Corp in the Bay Area of California.

Figure 5 shows the breakdown across both major releases in FIREFOX. FIREFOX appears to be very distributed even at the component level. Almost half of the components were categorized as spanning at least two sites and one third span multiple continents (i.e., at the world level).

Interestingly, there appears to be a dichotomy in geographic distribution. Modules are mostly either categorized as same site or worldwide.

Factor	Release 1.5 Pre-release		Release 1.5 Post-release		Release 2.0 Pre-release		Release 2.0 Post-release	
Intercept	**-1.14**	($\ll 0.01$)	**-0.78**	($\ll 0.01$)	**-1.22**	($\ll 0.01$)	**-0.69**	($\ll 0.01$)
LOC	**0.08**	($\ll 0.01$)	**0.08**	($\ll 0.01$)	**0.12**	($\ll 0.01$)	**0.07**	($\ll 0.01$)
CHURN	**0.11**	($\ll 0.01$)	**0.06**	($\ll 0.01$)	**0.12**	($\ll 0.01$)	**0.05**	($\ll 0.01$)
SPATIAL	**0.04**	($\ll 0.01$)	0.01	(0.21)	**0.03**	(0.01)	0.01	(0.05)
NATION	**0.62**	($\ll 0.01$)	**0.33**	($\ll 0.01$)	0.09	(0.53)	0.10	(0.31)
CONTINENT	-0.18	(0.19)	0.03	(0.68)	**0.23**	(0.03)	0.02	(0.76)
WORLDWIDE	**0.29**	($\ll 0.01$)	**0.16**	(0.02)	**0.28**	($\ll 0.01$)	0.04	(0.56)
ORGANIZATIONAL OWNERSHIP	**0.41**	(0.04)	0.21	(0.11)	0.31	(0.08)	0.19	(0.13)
adjusted R^2	**0.44**	($\ll 0.01$)	**0.31**	($\ll 0.01$)	**0.44**	($\ll 0.01$)	**0.23**	($\ll 0.01$)

Figure 3: The linear regression model results for effects of distributed development on quality in FIREFOX. Bold values are statistically significant and p-values are shown in parentheses

Levels of Distribution of Modules in FIREFOX

Rel.	Site	Nation	Continent	WorldWide
1.5	353 (53%)	72 (11%)	41 (6%)	202 (30%)
2.0	354 (50%)	43 (6%)	59 (8%)	249 (35%)

Figure 5: The number of modules at each level of distribution for FIREFOX.

We also investigated the differences between the modules categorized as same site and different continents. On average, modules categorized at the "Worldwide" level are larger, have more commits, and more distinct contributing developers. For instance both the average and median number of commits for modules developed on multiple continents were over three times higher than that of modules developed primarily in one site. A Mann-Whitney test showed that the difference between the two sets is statistically significant ($p \ll 0.01$).

This is in stark contrast to our previous study on Windows Vista development. Although geographically distributed binaries in Vista had, on average, more contributing developers, these binaries did not differ from their collocated counterparts in terms of size, complexity, or churn.

We therefore conclude that in answer to **RQ2**, *Large proportions of* FIREFOX *modules are developed at the same site and on different continents. However, same-site modules are smaller, have less commits, and less contributors.*

C. Effects of Distribution on Quality

Finally, we evaluate the effect that distribution has on software quality in FIREFOX. We used linear regression to measure the relationship of pre and post-release bugs in FIREFOX with the previously defined measures of distribution.

We used the same methodology when examining FIREFOX. The distribution of bugs in modules within FIREFOX is heavily right skewed; most plugins have between 5 and 20 postrelease bugs, but there is a long tail. We therefore applied a log transformation to the number of bugs prior to regression. In addition, LOC, CHURN, SPATIAL, TEMPORAL, ORGANIZATIONS were also transformed due to skew. We also found that LOC and COMPLEXITY were highly correlated leading to VIF values above 10. We also removed ORGANIZATIONS (correlated with TEMPORAL, SPATIAL, and CHURN) and TEMPORAL (correlated with ORGANIZATIONS and SPATIAL) due to high VIF. After applying these changes, the residuals on the resulting models followed a normal distribution, indicating that the results are valid.

We built models for pre and post-release defects for releases 1.5 and 2.0. The summary results of our regression models for FIREFOX are shown in **??**. Each column represents the coefficients and p-values (in parentheses) for statistical significance for one regression model. We use a standard cutoff of 0.05 for significance. Coefficients in bold indicate that the independent variable had a statistically significant relationship with defects. The sign of the coefficient indicates if defects went up (positive) or down (negative) as the value for the measure increased. Due to log transformations of dependent and independent variables, comparisons between coefficients are not straightforward. We therefore examine the direction and statistical significance of the geographic and organizational factors to determine if they are related to quality. In all cases where statistically significant, the measure of distribution had a positive relationship with defects. The one measure that is surprising is ORGANIZATIONAL OWNERSHIP, as we had expected that higher values would lead to fewer defects. Interestingly, bivariate spearman rank correlations of ORGANIZATIONAL OWNERSHIP varied from -0.32 to -0.54, indicating an inverse relationship. Once controlling for all other factors, however, this relationship reverses.

The results are somewhat mixed. In general, distribution measures are more significant during prior to release than after. Higher values of SPATIAL and worldwide distribution were associated with more defects in both prior to both releases. In contrast, the post-release defects in version 2.0 showed no significant relationship with any distribution measures after controlling for LOC and CHURN.

We therefore conclude for **RQ3** that *modules in* FIREFOX *that are geographically distributed show only a slight*

ECLIPSE Contributions by Organization		
Organization	**Commits**	**Percentage**
IBM	846500	91.6%
MIT	13174	1.4%
Self	11761	1.3%
BEA	2141	0.2%
Univ. of Victoria	1902	0.2%
Adv. Sys. Concepts	1729	0.2%
Prosyst	901	0.1%

Figure 6: Contributions to ECLIPSE from release 1.0 through release 3.3 by organization.

increase in failures and organizational distribution has no statistically significant effect in most cases.

VI. ECLIPSE RESULTS

We now present our results after performing a similar analysis on the ECLIPSE project

A. Organizational Distribution

Our data shows that IBM is the overwhelming contributor to ECLIPSE. Figure 6 shows the contributions to ECLIPSE across all releases. When broken down by release (not shown), IBM contributed 89% to 93% of the commits to each release, and in total, IBM contributed 91% of the commits to ECLIPSE. While it is generally recognized that IBM is the major backer of the ECLIPSE project, the level of development support (and clearly control) has not been quantified. The data is so extreme that no further quantitative analysis is needed to conclude that IBM contributes to ECLIPSE more than all other entities combined.

We therefore conclude for **RQ1** that ECLIPSE *is not organizationally distributed. IBM accounts for over 91% of the development that occurs in* ECLIPSE.

B. Development Sites

A disproportionate number of contributions to ECLIPSE come from a small number of geographical locations. Nearly half of the source code changes originate from one IBM site in Ottawa Canada and 75% can be traced back to just three sites. The graph in Figure 7 illustrates the percentage of contributions that were made from various sites. San Francisco and Cambridge are the only locations of those shown in the pie chart that are not completely represented by an IBM facility (there are a number of non-IBM sites aggregated into "Other"). Ottowa, Toronto, Portland, and Winnipeg all represent IBM facilities.

A small number of development sites account for the majority of source code contributions in ECLIPSE. *49% of contributions come from just one site and 76% of the contributions can be traced back to just three.*

When we look at the granularity of components (in ECLIPSE, plugins), the development is even less distributed. For instance, when we examine development for the JUnit code (a java testing framework) in the JDT component (Java

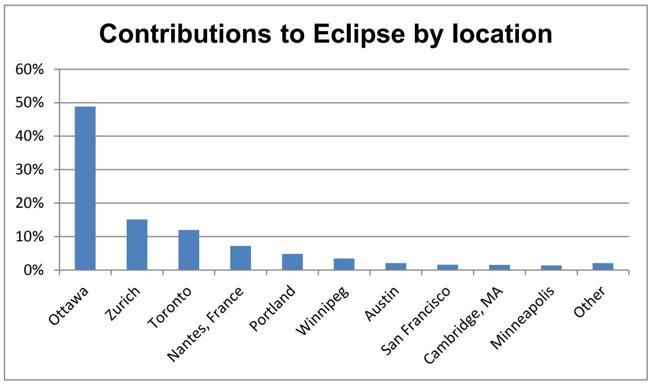


Figure 7: Contributions to ECLIPSE by location.

Development Tools) of ECLIPSE. The IBM site in Switzerland accounts for 96% of the commits in JUnit (3,094 total commits), and the other 4% of the commits come from Ottawa, Cambridge, Winnipeg, and Portland (all IBM sites).

We found that as a whole system ECLIPSE is distributed worldwide, but is dominated by just three sites on two continents. However, when examined at the level of individual components, they are not very geographically distributed at all. The numbers of plugins that were not categorized as same site were quite small. Figure 9 shows the breakdown across six major releases in ECLIPSE. There were few cases where a component is distributed across multiple continents; only 1% of the plugins in release 3.3 had non-trivial contributions coming from both Europe and North America.

Similar to our organizational results for ECLIPSE, we conclude for **RQ2** that *the vast majority of plugins in* ECLIPSE *are not geographically distributed.*

C. Effects of Distribution on Quality

As with FIREFOX, we used linear regression to measure the relationship of pre and post-release bugs in ECLIPSE with levels of distribution. Given the low number of distributed plugins (some releases had only 4), some of the regression models lacked enough samples for statistical power.

The distribution of defects in plugins in ECLIPSE is heavily right skewed, meaning that while most plugins have between 10 and 50 defects, there is again a long tail and a few plugins have hundreds of defects. When testing for multicollinearity, we had to remove ORGANIZATIONAL OWNERSHIP and TEMPORAL. In order to fit residual normality assumptions, we performed a log transformation on LOC, CHURN, and ORGANIZATIONS. We summarize the results of our regression analysis in Figure 8. Statistically significant values at 0.05 are in bold and at 0.10 are in italics. We also include adjusted R^2 values for the models.

Again, the results are somewhat mixed. Each measure shows a statistically significant effect in some releases and not in others. Some releases don't show a significant effect of geographic distribution on software quality at certain levels. However, often this is due to small sample sizes. For instance, the 3.3 release shows no effect of worldwide distribution on

	2.0		2.1		3.0		3.1		3.2		3.3	
Factor	**Pre**	**Post**	**Pre**	**Post**	**Pre**	**Post**	**Pre**	**Post**	**Pre**	**Post**	**Pre**	**Post**
LOC	**0.29**	0.18	**0.23**	**0.24**	**0.37**	**0.30**	**0.28**	**0.24**	**0.29**	**0.14**	**0.21**	**0.29**
Churn	**0.54**	**0.35**	**0.44**	**0.30**	**0.32**	**0.24**	**0.40**	**0.24**	**0.44**	**0.41**	**0.41**	**0.20**
Nation	-0.42	-0.05	**1.22**	**0.98**	*1.41*	**1.24**	**1.46**	*0.75*	0.42	0.28	0.16	0.30
Continent	**2.16**	**1.54**	**1.28**	**1.30**	**1.02**	**0.83**	**1.34**	0.35	0.24	0.04	0.21	-0.18
Worldwide	0.62	*1.54*	0.21	-0.31	-0.14	0.51	0.59	**1.00**	0.11	0.50	0.50	*0.93*
Organizations	0.74	1.04	*0.80*	0.43	-0.22	0.12	**-0.74**	0.04	*-0.70*	**-1.03**	*-0.64*	-0.42
adjusted R^2	**0.82**	**0.65**	**0.69**	**0.64**	**0.61**	**0.60**	**0.69**	**0.55**	**0.73**	**0.68**	**0.67**	**0.57**

Figure 8: The linear regression model results for effects of distributed development on quality in Eclipse.

Levels of Distribution of Plugins in Eclipse				
Rel.	**Site**	**Nation**	**Continent**	**Worldwide**
3.3	138 (86%)	9 (6%)	11 (7%)	2 (1%)
3.2	138 (81%)	7 (4%)	22 (13%)	4 (2%)
3.1	133 (87%)	7 (5%)	8 (5%)	5 (3%)
3.0	105 (78%)	3 (2%)	21 (16%)	5 (4%)
2.1	73 (78%)	4 (4%)	7 (8%)	9 (10%)
2.0	55 (76%)	6 (8%)	9 (13%)	4 (3%)
All	642 (82%)	36 (5%)	78 (10%)	29 (3%)

Figure 9: The number of modules at each level of distribution for Eclipse.

quality, but that is likely because although the distributed components did have more pre- and post-release bugs, only 2 plugins were distributed worldwide. This represents only 1% of the plugins in 3.3.

Although not shown in the figure, the magnitude of the increase in bugs was also significant. In all cases where distribution was significant, the average number of bugs for the distributed plugins was at least 50% more than the collocated plugins; *in many cases they had twice as many bugs or more.* Given the small number of plugins that were distributed worldwide, this does not always represent a statistically significant result.

Thus our findings for **RQ3** are that *all measures of geographic and organizational distribution increase failures, but the effects are not consistent across releases.*

VII. Threats to Validity

One threat to construct validity is that we may not capturing all defects for these projects. We only include bugs that have been closed because the location of the source of an unclosed bug is impossible to determine without manual inspection. We argue that fixed bugs are the most important to the project (being important enough to warrant being fixed), and are a reasonable proxy for software quality. A prior study [29] found that there is some bias in Eclipse bug data relative to bug severity and experience of the bug closer. We have not examined potential bias in organizational or geographic dispersion.

A potential threat to external validity is that we do not capture the ecosystems surrounding both Firefox and Eclipse. There are a number of external plugins that exist for Eclipse and Firefox that were developed by separate open source and commercial organizations. For Eclipse we only study the core platform and plugins. This represents the functionality that a developer gets when downloading one of the standard versions (we include all plugins that occur in all standard versions, as each release of Eclipse has multiple downloadable sets of plugins) Eclipse from the project web site. Similarly, we do not include any Firefox addons in our analysis. While this means that we do not capture *all* development relative to these projects, this decision was based on the fact that we are interested in the core product. Developers of third-party plugins and addons may have little or no similarity at all.

The impact of organizational and geographic factors on improvement of the baseline failure models is related to the features used in the original models. We had to make decisions regarding what metrics to include in our baseline models and included size and churn because these have been shown to consistently have a strong relationship to failures [25], [30]. Other factors were either difficult to compute (firefox is written in many languages so any source analysis would be difficult) or have shown inconsistent results in different domains (*e.g.*, number of contributors to a software entity [31], [32]).

VIII. Conclusion

We have presented the first study that characterizes both organizational and geographically distributed development and examined the effects of these factors on software quality in two large scale, successful open source projects.

Firefox development is quite distributed both geographically and organizationally, although approximately half of all commits come from Mozilla Corp. organizationally and half come from Californa geographically. Only half of all modules have 75% of their changes originating from one city. Modules that are geographically distributed tend to be larger, more complex, and have more contributors. They also exhibit more defects even when controlling for size and churn.

Eclipse differs in that it has very low organizational distribution. At the system level, Eclipse is also geographically distributed, but this is due to a small number of IBM sites scattered around the world. At the component level, there is very little distribution (i.e., almost every component is developed largely in one location). Geographically distributed components have many more defects that those that don't (though not always statistically significant), although the effects of distribution have lessened in later releases. Those components that are organizationally distributed have less defects in the few cases that are statistically significant.

This paper combined with previous studies of geographically and organizationally distributed projects serves to build a body of knowledge regarding the growing field of distributed software development. Clearly, there is continued work to be done in order to understand the contexts, processes, and individual factors that influence distributed development and its relationship to important outcomes. This is an area that needs continued study, as distributed development is becoming an increasingly used approach and there is still much to be learned about how they differ from collocated development.

REFERENCES

[1] E. S. Raymond, "The revenge of the hackers," in *Open Sources: Voices from the Open Source Revolution*, C. DiBona, S. Ockman, and M. Stone, Eds. O'Reilly and Associates, 1999. [Online]. Available: http://www.oreilly.com/catalog/opensources/book/raymond2.html

[2] W. Scacchi, "Understanding requirements for developing open source software systems," *IEE Proceedings - Software Engineering*, vol. 149, no. 1, 2002.

[3] T. Zimmermann, A. Zeller, P. Weissgerber, and S. Diehl, "Mining version histories to guide software changes," *Software Engineering, IEEE Transactions on*, vol. 31, no. 6, pp. 429–445, 2005.

[4] A. Mockus, J. D. Herbsleb, and R. T. Fielding, "Two Case Studies of Open Source Software Development: Apache and Mozilla," *ACM Trans. on Software Engineering and Methodology*, vol. 11, no. 3, July 2002.

[5] P. Rigby, D. German, and M.-A. Storey, "Open Source Software Peer Review Practices: A Case Study of the Apache Server," in *Proceedings of the International Conference on Software Engineering*, 2008.

[6] E. Shihab, Z. M. Jiang, and A. E. Hassan, "On the use of Internet Relay Chat (IRC) meetings by developers of the GNOME GTK+ project," in *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories*. IEEE Computer Society, 2009.

[7] D. S. Pattison, C. Bird, and P. T. Devanbu, "Talk and work: a preliminary report," in *Proceedings of the Working Conference on Mining Software Repositories*, 2008.

[8] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Chapels in the bazaar? latent social structure in oss," in *16th ACM SigSoft International Symposium on Foundations of software engineering*, 2008, pp. 24–35.

[9] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy, "Does distributed development affect software quality?: an empirical case study of Windows Vista," *Communications of the ACM*, vol. 52, no. 8, 2009.

[10] F. Shull, J. Carver, S. Vegas, and N. Juristo, "The role of replications in empirical software engineering," *Empirical Software Engineering*, vol. 13, no. 2, pp. 211–218, 2008.

[11] D. Spinellis, "Global Software Development in the FreeBSD Project," in *Proceedings of the 2006 international workshop on Global software development for the practitioner*, 2006.

[12] R. Tang, A. Hassan, and Y. Zou, "Techniques for identifying the country origin of mailing list participants," in *16th Working Conference on Reverse Engineering*, 2009.

[13] G. Robles, S. Duenas, and J. Gonzalez-Barahona, "Corporate involvement of libre software: Study of presence in debian code over time," *Open Source Development, Adoption and Innovation*, 2007.

[14] G. Robles and J. Gonzalez-Barahona, "Geographic location of developers at sourceforge," in *Proceedings of the 2006 international workshop on Mining software repositories*. ACM, 2006, pp. 144–150.

[15] M. Cataldo and J. Herbsleb, "Factors leading to integation failures in global feature oriented development: An empirical analysis," in *Proceedings of the 33rd international conference on Software engineering*. ACM, 2011.

[16] N. Ramasubbu, M. Cataldo, R. Balan, and J. Herbsleb, "Configuring global software teams: A multi-company analysis of project productivity, quality, and profits," in *Proceedings of the 33rd international conference on Software engineering*. ACM, 2011.

[17] M. Cataldo and S. Nambiar, "On the relationship between process maturity and geographic distribution: an empirical analysis of their impact on software quality," in *Proceedings of ACM SIGSOFT symposium on The foundations of software engineering*, 2009.

[18] N. Nagappan, B. Murphy, and V. Basili, "The influence of organizational structure on software quality: an empirical case study," in *Proceedings of the 30th international conference on Software engineering*, 2008.

[19] M. Cataldo, P. Wagstrom, J. Herbsleb, and K. Carley, "Identification of coordination requirements: implications for the Design of collaboration and awareness tools," *Proc. of the 20th conference on Computer supported cooperative work*, 2006.

[20] G. Boetticher, T. Menzies, and T. Ostrand, "PROMISE Repository of Empirical software engineering databases." West Virginia University, Dept. of Computer Science, 2007. [Online]. Available: http://promisedata.org/repository

[21] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" in *Proceedings of the 2nd International Workshop on Mining Software Repositories*, 2005.

[22] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *Proceedings of the 3rd International Workshop on Mining Software Repositories*, 2006.

[23] Billy Biggs, Personal Interview, August 2009.

[24] M. O'Leary and J. Cummings, "The spatial, temporal, and configurational characteristics of geographic dispersion in teams," *Mis Quarterly*, vol. 31, no. 3, pp. 433–452, 2007.

[25] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in *Proceedings of the 28th international conference on Software engineering*, 2006.

[26] P. Allison, *Multiple regression: A primer*. Pine Forge Pr, 1999.

[27] G. Jay, J. Hale, R. Smith, D. Hale, N. Kraft, and C. Ward, "Cyclomatic complexity and lines of code: empirical evidence of a stable linear relationship," *Journal of Software Engineering and Applications*, no. 2, 2009.

[28] E. Weyuker, T. Ostrand, and R. Bell, "Using developer information as a factor for fault prediction," in *Proceedings of the International Workshop on Predictor Models in Software Engineering*, 2007.

[29] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu, "Fair and Balanced? Bias in Bug-Fix Datasets," in *Proceedings of the ACM SIGSOFT Symposium on The Foundations of Software Engineering*. ACM, 2009.

[30] R. Bell, T. Ostrand, and E. Weyuker, "Does measuring code change improve fault prediction?" in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*. ACM, 2011, p. 2.

[31] E. J. Weyuker, T. J. Ostrand, and R. M. Bell, "Do too many cooks spoil the broth? using the number of developers to enhance defect prediction models," *Empirical Softw. Engg.*, vol. 13, no. 5, pp. 539–559, 2008.

[32] C. Bird, N. Nagappan, B. Murphy, H. Gall, and P. Devanbu, "Don't touch my code! examining the effects of ownership on software quality," in *Proceedings of the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, 2011.