Investigating the Geography of Open Source Software through Github

Yuri Takhteyev
University of Toronto
Faculty of Information
yuri.takhteyev@utoronto.ca

Andrew Hilts
University of Toronto
Faculty of Information
andrew.hilts@utoronto.ca

ABSTRACT

The paper presents an empirical study of the geography of open source software development that looks at Github, a popular project hosting website. We show that developers are highly clustered and concentrated primarily in North America and Western and Northern Europe, though a substantial minority is present in other regions. Code contributions and attention show a strong local bias. Users in North America account for a larger share of received contributions than of contributions made. They also receive a disproportionate amount of attention.

Author Keywords
open source software, geography, distance, collaboration, empirical study, global software development

ACM Classification Keywords
D.2.9 [Software Engineering]: Management: Programming teams; H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces: Computer-supported cooperative work. K.1 [Computing Milieu]: The Computer Industry

INTRODUCTION

Location has always been a crucial factor in organization of work. The numerous reasons why specialized industries cluster in particular places, for example, was already analyzed by Marshall at the end of the 19th century (Marshall, 1890/1927). The rise of modern telecommunication technologies, however, has lead many to ask whether place and distance would remain important for work, especially for knowledge work that is assumed to involve primarily manipulation of information. Much of the research since the 1970s has suggested that place is likely to continue to be important (Short et al, 1976; Olson & Olson, 2000; Bradner & Mark, 2002). Despite the challenges that distance presents for cooperative work, however, remote collaboration has become a daily reality for many knowledge workers. This paper itself has been written largely by the co-authors while they were separated by over 8,000 km. It is thus important for us to base our discussions of the feasibility and challenges of global collaboration on empirical studies that measure the extent of such collaboration. In this paper, we look at the geographic dimensions of open source software development using a dataset derived from Github.com, a popular open-source project hosting site. We explore both the geographic distribution of Github users and the relationship between distance and the formation of collaborative ties.

Software development presents a particularly interesting domain in which to analyze the interaction between distance and collaboration. It is often understood as a field in which distance matters less, due to the largely immaterial nature of software work. At the same time, software production is vastly concentrated; California's Silicon Valley is often used as a text-book example of a regional industry cluster. *Open source* software development is often imagined to be even more independent of distance, since its developers can be assumed to be free to start their projects wherever they are and to contribute to any projects around the world. A number of well-known examples seem to give support to this global imaginary: Linux originated in Helsinki, and the Ruby programming language was developed in Japan, Ubuntu hails from South Africa. These examples become more complicated, of course, once we consider that the author of Linux now lives in the United States, that Ruby came to prominence through "Rails", a system built in Chicago, and that Ubuntu is developed by a company based in London.

EARLIER WORK

Locating OSS Developers

Studies about OSS developers in general are often limited in reliability because the complete population of developers cannot be easily defined or sampled. Prior studies have typically relied on snowball surveys of developers (Ghosh et al, 2005), case studies of specific projects (Spinellis 2006; Tang et al, 2006), or looked at online software

The paper is currently under review.
This version is from August 6, 2010.

repositories (Dempsey et al 1999; Robles & Gonzalez-Barahona, 2006; Gonzalez-Barahona et al, 2008; Von Engelhardt et al 2010). Survey methods, for example as employed by Ghosh et al (2005), have the advantage of allowing the researcher to select questions to be asked of the participants, but raise particularly difficult questions of generalizability. A case study examining a single or small numbers of projects can be useful for in-depth analysis of intra-project work dynamics (e.g., Ducheneaut, 2005; Spinellis, 2006; Tang et al. 2006) and can be easier to interpret, since researchers can rely on all that is known about the particular project to discuss what this project is (or is not) representative of. For broader generalization, however, researchers need to look at a multitude of projects.

Analyzing data from services offering hosting to a large number of open source projects has been a popular solution. While such studies retain certain characteristics of case studies (each hosting service has its own unique history and community), they can offer more generalizability than studies of a single project or small set of them, while being somewhat easier to interpret than snowball surveys.

SourceForge.org has been particularly important in advancing research on OSS development (see, for example Robles & Gonzalez-Barahona, 2006; Crowston et al., 2006; Gonzalez-Barahona et al., 2008; Subramanyam & Xia, 2008; Von Engelhardt et al, 2010). Its popularity among researchers is closely linked with its popularity among developers. Throughout the past decade, SourceForge was widely seen as *the* place for hosting open source software projects and was thus assumed to be more representative of the total universe of OSS developers than other, smaller, systems. The global representativeness of SourceForge is questioned by scholars such as Subramanyam & Xia (2008), who complement their studies by an investigation of an Indian and a Chinese community websites, Sarovar.org and HuiHoo.org. Unfortunately, studies involving multiple hosting sites create challenges for aggregating data and avoiding double-counting developers who are involved with multiple sites. Studying "global" hosting services as SourceForge, in contrast, provides us with readily comparable project data as well as a defined population of developers.

A significant and recognized limitation of using SourceForge in particular as a case study for investigating the geography of OSS development is that the locations of developers are not explicitly stored (Robles & Gonzalez-Barahona, 2006; Gonzalez-Barahona et al., 2008; Von Engelhardt et al, 2010). Therefore, researchers have relied on a number of different methods for inferring users' location. Locations can be inferred to some exntent from the top-level domains (TLDs) of developer e-mail addresses (eg., .ca, .jp, .br). E-mail address TLDs obtained from both SourceForge and project mailing lists are widely employed as the base data source for developer locations (Dempsey et al, 1999; Robles & Gonzalez-Barahona, 2006; Gonzalez-Barahona et al., 2008; Von Engelhardt). The researchers all recognize that this method is limited due to the common occurrence of generic TLDs such as .com, .net and .org, users of which may not be clearly associated with a country. In our own sample of Github users, accounts reporting their location in the ten most commonly countries other than the United States use an email address in their country's TLD with frequencies that vary from 2% (China) to 37% (Germany), with the frequency of 10-20% being most common.[1]

Some researchers have looked for ways to make use of timezones associated with a user profile in addition to proportional statistical inferences for users without a timezone (Robles & Gonzalez-Barahona, 2006; Gonzalez-Barahona et al., 2008). More recent work (e.g., Von Engelhardt et al, 2010) has avoided this problem by making use of a research database that provides IP addresses from which the users log into SourceForge. (This dataset is not publicly available and requires a signed agreement.)

While SourceForge has provided valuable data about the OSS community, the emergence of new systems for version control such as git and bazaar in the recent years, combined with SourceForge's slow adoption of such services (as well as a number of other features) has lead to a proliferation of several alternative hosting sites. In this paper, we look at one of such sites, Github.com, which has recently grown in popularity and does not feature some of the limitations discussed above, providing richer data about OSS development activities.

Results from the studies mentioned above show that developers are predominately located in North American and European countries, particularly in the United States. (As Subramanyam & Xia point out, this could in part reflect that those studies surveyed developers in those counties or looked at "Western" hosting services such as SourceForge.) An early study of Linux development archives found a heavy European presence in the total number of contributors (Dempsey et al, 1999). A survey by Ghosh et al. (2005) found that French, American and German residents were respectively the most numerous developers. Later studies, however, presented different results. A project-specific study by Tuomi (2005) found that most developers identified with the United States (37%), followed by Germany (17%), the United Kingdom (8%), and Canada (6%). A study of SourceForge (Robles and Gonzalez-Barahona, 2006; Gonzalez-Barahona et al. 2008) similarly reported that most registered developers were located in the USA (36%), Germany (8%), the UK (5%) and Canada (4%). In another study of SourceForge, which measured not merely registered, but active developers that had contributed to a project within a year of the study, the

[1] Needless to say, very few (<1%) of the US-based users use a ".us"

concentration of developers in the United States in particular was even greater: the United States accounted for 44%, followed by Germany (9%), the UK (5%), and Canada (4%) (Von Engelhardt et al, 2010).

At a continental level, most developers of the FreeBSD project were found to reside in North America (46%) and Europe (39%), with small clusters appearing in Asia (10%), Australia (2.5%), South America (1.6%), and Africa (0.8%) (Spinellis, 2006). This ordering is roughly consistent with the earlier study of SourceForge that reported registered users as being located in North America (41%), Europe (30%), Asia (11%), Oceania (4%),[2] South America (3%) and Africa (1%) (Robles and Gonzalez-Barahona, 2006; Gonzalez-Barahona et al. 2008).

These results indicate that the most frequently occurring national residences of registered SourceForge users are very similar to the top countries by GDP, with the notable exceptions of Japan performing lower and Canada and Australia performing higher (Robles and Gonzalez-Barahona, 2006; Gonzalez-Barahona et al., 2008). Furthermore, an even greater proportion of active SourceForge developers reside in OECD countries (Von Engelhardt et al, 2010). However, the same study reported that once Internet access was taken into account, the ratio of countries' Internet users to OSS developers were less disparately distributed, though still uneven.

Ties and Distance

Uneven global distributions of open source software developers likely has many reason. One simple factor, for example, is the variation in Internet access in different parts of the world (Gonzalez-Barahona et al, 2008; Von Engelhardt et al, 2010), as well as the broader differences in economic development. An important contributing factor, however, is likely the *locality* of the open source practice. Since open source software development has its roots in specific places, we would expect it to remain disproportionately concentrated in such places to the extent that distance presents a barrier to collaboration and to the reproduction of the practice elsewhere.

Olson & Olson (2000) review a substantial body of research that shows the many different ways in which distance can hinder collaboration, even in the presence of modern communication technologies. Distance can impede collaboration by preventing face-to-face interaction, introducing differences of context and culture, and often by introducing a difference of time zone. Studies of software development projects have shown that such projects are not immune to the effect of distance (Carmel and Agarwel, 2001; Holmström et al, 2006; Herbsleb, 2001; Del Rosso, 2009). However, a number of methods have been proposed for reducing the effects of distance in software development, such as the division of software into more autonomous, loosely-coupled modules (Olson & Olson, 2000; Carmel and Agarwel, 2001; German, 2003; Gutwin, 2004). Cultural distance can be also reduced by collaborating with international partners with similar language or cultural values (Carmel and Agarwel, 2001). Carmel and Agarwel predict that "projects will increasingly look like a global virtual archipelago with several separate clusters of colocated professionals sprinkled with dispersed individuals working remotely" (p. 29).

Studies of open source software developers have also demonstrated the effects of distance, though many studies have also pointed out the commonality of long-distance collaboration in OSS. Spinellis (2006) compared the geographic distance between mentor-mentee pairs in the FreeBSD project and found that these pairs were more likely to be established within the same geographic area than regular collaborating developers. However, the author also notes that such mentorship relationships are also found across continents. Furthermore, the study reported the average distance between all FreeBSD contributors and that of closely collaborating contributors are roughly equal, at around 6,500 km, suggesting that collaborative activity in that project was not heavily constrained by geographic distance. Tang et al (2009) measured the 'spread' (in hours) between the timezones of an OSS project's mailing list discussants. They found that discussion initiators from outside of North America, Canada and the EU experience high response delays. Distance has also been found to be a factor in studies of other collaborative communities such as Wikipedia and Flickr (Crandall et al, 2009, Hecht and Gergle, 2010).

Degrees of Participation in OSS Projects

While in theory anyone can contribute to an open source project, permissions and respect within the development community are not evenly distributed. Crowston & Howison (2005) conceptualized different degrees of involvement in an open source development project as a layered onion. On the outside of the project operate the software users, some of whom also operate at the project's periphery as mailing list subscribers. The next layer of involvement includes bug reporters and fixers, while the core of the project consists of the regular developers who write and commit the bulk of the source code. Indeed, many scholars have observed the tendency of OSS projects to feature a core group of developers plus a wider peripheral community (see Crowston et al. 2006). For instance, Mockus et al.'s widely-cited study found that a small core of developers (15%) contribute a large majority (83%) of modification requests to Apache's source code (Mockus et al. 2000). A later case study also noted a core/periphery structure of 15 developers contributing 57% of new code in the FreeBSD project, albeit with a less extreme concentration than Mockus et al's findings (Dinh-Trong and Bieman, 2005).

[2] We assume that in this case Oceania includes Australia and New Zealand.

The roles and responsibilities afforded to different areas within development hierarchies each contribute to the the project's overall culture, community and evolving software product. Indeed, Kuk found that the KDE developer mailing list exhibits a concentration of discussion topics surrounding a core group of mailing list participants (Kuk, 2006). Kuk argues that this participation inequality is essential for knowledge sharing and management of OSS development, which is consistent with the general models of learning, such as Lave & Wenger's (1991) notion of "legitimate peripheral participation" in communities of practice. Peripheral participants can also have an impact on projects' development by providing ideas, even if such ideas are filtered by the core group (Barcellini et al. 2008).

Various authors have argued that the core group of developers in an open source project tend to be more homogeneous than those occupying the periphery. This homogeneity is expressed in terms of shared goals and mental models in relation to the specific software project and in terms of perceived technical expertise (e.g., Ducheneaut, 2005). Demazière et al. (2007) case study found that a core of contributors to the project had a shared "community of experience" (p. 11), while an intermediate circle of contributors had more diverse involvement rationales for participation. This shared sense of community and thus low sociocultural distance could be encouraged by the fact that developers are self-selected and thus perceive an alignment between their own goals and the project's goals (Lundell et al., 2002).

The different degrees of participation become important for studies of the geography of open source software development. For example, we can expect that more central forms of participation (e.g., maintaining a project) can be associated with central places, while peripheral forms of participation (e.g., filing bug reports and making smaller code contributions) may be more prevalent in peripheral locations. For example, studies that measured not merely registered, but active developers have often reported higher concentration of developers in the United States than those that attempted to count all developers.

DATA SOURCE AND METHOD

Github

In this paper we look at geographic dimensions of development on Github, a website that provides hosting for open source software projects using git. While Github now provides a range of features, including bugtracking, blogs, and project wikis, its defining feature has been version control using git. Git is a version control system developed in 2005. Together with other "distributed" version control systems (such as *bazaar* and *mercurial*), git makes it easier to merge once-forked repositories, thus encouraging frequent code-forking and enabling a more decentralized development model (Torvalds 2007, Bird et al, 2009). The rapid growth in popularity of distributed version control has lead to appearance of sites providing hosting of software repositories using such systems. In the recent years, Github emerged as the most popular hosting system focused on git. Github now hosts such popular projects as Ruby on Rails, CakePHP, JQuery and curl and has recently claimed to be hosting one million repositories (Holman, 2010).[3]

GitHub users are invited to specify their geographic location in their individual profiles. While not all users provide their location, many do. The locations provided vary in precision and seriousness, though the overwhelming majority of those who specify a location provide a description that can be easily interpreted as referring to a particular place (see below). GitHub therefore offers a public dataset with self-reported user location data, avoiding the problems commonly associated with analysis of SourceForge data.

GitHub also borrows a number of features from recently popular "social media" systems such as Twitter. A Github user can choose to "follow" another user, essentially subscribing to an update stream of the followed user' activity, which is displayed when a user logs into the website. Similarly, users can choose to "watch" other users' repositories. The system also tracks code contributions between users, making it easy to identify who has made contributions to whose repositories.[4] Github thus provides researchers with a collection of relational data, measuring several different relationships (following, watching, making a contribution) and making it possible to investigate how each of those relationships is affected by place.

Data collection

We collected our data through Github's public API, which offers the same data as available on the Github's website but presents it in a structured format for simpler processing. The data were collected from May to July of 2010. The data collection followed a recursive procedure. We started with a single account, belonging to one of Github's founders. We then identified accounts connected to this user, then looked for accounts connected to the newly found ones, repeating this procedure until we achieved closure. New accounts were identified through the four kinds of connections mentioned in the previous section: (1) those that *follow* accounts collected earlier, (2) those *followed by* the accounts collected earlier, (3) those whose repositories were being "watched" by accounts collected earlier, (4) those who had made code contributions to the repositories watched by accounts collected earlier. (In the case of following, Github makes it easy to retrieve lists of users

[3] This number includes forked repositories.

[4] Github's tracking of contributions depends on the user providing the site with the same email address as they use in their git client. However, the site provides the users with an incentive to use the same address in both places, since the users can see when their contributions are not being properly associated with their account.

who follow and are being followed by a given account. In the cases of watching and contributions, the system only allows one-way queries: a list of repositories watched by a user and a list of contributors to a repository.)

The process reached closure at 70,414 accounts – at this point none of the accounts in this set were connected to any of the accounts outside. This number is smaller than the number of registered users advertised by Github at the time – around 250,000. We believe that this is likely due a large number of isolated accounts.

Relations

In addition to collecting the basic account records for those 70,414 users, we also collected the relationships between them, obtaining pairs linked by directed relationships of **following**, **repository-watching** and **code contributions**. In the case of following, the ties were binary. In the case of repository watching and code-contributions the ties have weights: user X can make just one contribution to Y's repository, or a large number of them. Similarly, user X can watch just one or many repositories maintained by Y. For some of our statistics below we count the number of unique pairs while for others we count the total number of contributions and "watchings."

Git differs in an important way from centralized revision control systems such as Subversion in that one can easily "fork" and later re-synchronize repositories. As a result, Github contains a substantial number of forked repositories. While such forking is potentially an interesting phenomenon for investigation, we limited our analysis of code contributions to only *original* repositories, ignoring all forks. The reason for this is that contributions to forked repositories often do not imply any direct connection between the contributor and the author of the forked repository: the presence of a contribution by user X in the repository of Z may simply mean that Z's repository is a copy of Y's, to which X made a contribution. X and Z may have no relations. While contributions to the original (unforked) repositories may be similarly made indirectly (X could contribute to Z's copy of Y's repository, after which Y would pick up the contribution from Z's repository), we believe such indirect flow is meaningful, and in some ways is the *raison d'être* of distributed source control: users often make contributions to forked repositories with the intention that such contributions would eventually propagate to the original repository.

Geocoding

Of the 70,414 accounts, 32,503 (46%) specified some value for location. Nearly all of those descriptions (31,977, or 45% of all accounts) referred to an actual location, identifying at least the country. A substantial majority (26,509 or 38% of all accounts) further identified location at the level of a city or some other place with an area of up to 25,000 km^2 – the size of a large metropolitan region.[5] The share of accounts identifying a specific location was higher for the more active accounts. For example, for accounts associated with the 10,000 most watched repositories, as many as 82% provided a specific location.

Tokyo, Japan	Tokyo, JP	tokyo japan
Tokyo	Japan/Tokyo	東京都千代田区, 日本「Tokyo,Japan」
tokyo	Japan Tokyo	Tokyo, Ulanbaatr
Tokyo Japan	Tokyo JP	Tokyo Shibuya
Tokyo/Japan	Tokyo, Japan.	Tokyo, Setagaya
TOKYO	tokyo, Japan	Tokyo, JPN
東京	tokyo.japan	

Table 1: User-supplied locations descriptions for Tokyo.

The users employed a total of 8,767 unique descriptions to specify their locations. The diversity of methods and conventions for describing locations presented a substantial problem for geocoding. Table 1 shows some examples of the ways users identified their location as being in Tokyo. As indicated by those examples, users may or may not specify the country and have different conventions as to the order in which the country, the city and other geographic units are listed. Ignoring such variation runs a substantial risk of under-counting users from outside the United States. On the other hand, searching location strings for names of cities can introduce false positives and again bias the data towards larger cities, for example by counting places like London, Ontario towards London, England. Some of the popular geocoding APIs similarly introduced a substantial number of false positives.

To maximize precision, we used a combination of methods to code the data. First, we attempted to parse the location into (1) a city and country pair, (2) a city and state/province pair, or (3) a city, state/province and country triplet, allowing for a variety of ordering and delimiters. (When looking for city-province-country triplets, we checked immediately that such a province exists in the corresponding country.) In cases where such parsing was successful, we used Geonames database[6] to check if a city by such a name in fact exists in the stated province and country. If such a city was found, we accepted it as the intended location and associated the account with geo coordinates provided by Geonames. Location that could not be parsed or were parsed into a pair or triplet that could not be found in Geonames, were then run through Yahoo's GeoAPI. We found that Yahoo's GeoAPI identified the

[5] The 25,000 cutoff was introduced for the sake of consistency. If we were to include "San Francisco Bay Area" as a "specific" location, we wanted to make sure to also include other areas of comparable size, such as "Wales."

[6] http://geonames.org/

locations correctly in the majority of cases, but produced a very large number of false positives. For this reason, the results were checked and corrected by hand, then verified using the Geonames database. Overall, we decoded 57% of the locations using the fully automated method, while 43% of descriptions required some degree of manual processing.

Clustering

Locations specified by the users referred to units of rather different size, which in some cases overlapped each other. For example, some users identified their location as "Brooklyn, NY" while others placed themselves in "New York, NY." (This problem is common with self-reported locations. Many statistics for Twitter show "Brooklyn, NY" as a separate city.) To avoid ad-hoc solutions to this problem (e.g., manually assigning Brooklyn to New York), we merged the locations into regional clusters, using an iterative procedure, replacing nearby locations with a single point located at the average of their geocoordinates, weighted by the number of observations at each point. For example, the 238 observations labeled "New York, NY" and 180 observations labeled "Brooklyn, NY," would be replaced by a single point, now representing 418 accounts, located between the geo-coordinates originally assigned to "New York" and "Brooklyn" (238/418th of the way from the original New York point to the original Brooklyn Point). This procedure resulted in 679 **local clusters**. The average distance between the original position of a location and the center of the cluster to which it was assigned was 22 km, with a standard deviation of 27 km. The maximum distance was 158 km. The resulting clusters varied substantially in the number of observations that were merged into them. For example, a cluster corresponding to San Francisco Bay Area included 103 unique points (representing 1985 accounts), while 38% of the clusters consisted of a single location.

RESULTS

Users and contributions by region

Consistent with earlier studies of open source software developers, the United States accounts for the largest share of the registered user accounts by far: 39%. The remaining 61% of users are spread between a large number of countries, none of which accounts for more than 7%. (The second largest country is the United Kingdom, at 7%, followed by Germany at 6%. Both numbers are consistent with the earlier literature.)[7] We present country statistics for the top ten countries in table 2.

Country	Share of users	Share of contributors	Share of conributions
USA	38.6	38.7	43.1
UK	7.3	7.7	6.5
Germany	5.9	6.2	6.1
Canada	4.2	4.3	4.3
Brazil	4.2	3.6	2.2
Japan	3.9	3.9	5.2
France	3.0	3.2	3.2
Australia	2.9	3.1	3.1
Russia	2.2	2.3	2.2
Sweden	2.0	2.2	2.3

Table 2: Github participation by country (%).

Since countries vary substantially in size, we avoid further discussion of results by country and instead focus on two levels: macro-regional and local, only mentioning specific countries when regional statistics appear substantially influenced by a single country.

For macro-regional analysis we merged country data into sub-regions as defined by the UN Statistics Department.[8] Due to the large number of sub-regions, most of which have rather few data points, we further merged sub-regions within the same continental region to the extent that they appeared to have similar profiles. In particular, we merged all sub-regions within Africa and Asia and merged all sub-regions in the Americas except for North America into "Latin America," keeping "North America" separate due to its obviously different profile from the other American sub-regions. Finally we merged the four sub-regions of Europe into two: "Western and Northern Europe" (further "W&N Europe") and "Eastern and Southern Europe" (further "E&S Europe"). In both cases we merged sub-regions that seemed to be quite similar in terms of the statistics discussed below.

As shown in table 3 and the outer ring of figure 1, North America and W&N Europe account respectively for 43% and 26% of the users. The remaining regions account for 11% or less each. The North America's remains about the same if we only count users for whom we registered at least one code contribution, but rises to 48% when we count at the total number of contributions associated with accounts in North America. This gain comes at the expense of each of the other regions except for Australia and New Zealand. The loss is most substantial for Latin America, which accounts for 6.4% of registered user accounts but only for

[7] While our numbers are consistent with the counts of registered open source developers by Robles & Gonzalez-Barahona (2006) and Tuomi (2005), they differ substantially from those reported by Ghosh et al (2005), which gave substantially higher numbers for European countries and in particularly for France (15%).

[8] See http://unstats.un.org/unsd/methods/m49/m49regin.htm. It needs to be noted that UNSD's sub-regions differ somewhat from colloquial usage of the regional names. In particular, "North America" includes only United States and Canada, but not Mexico.

	North America	W&N Europe	E&S Europe	Asia	Latin America	Australia and New Zealand	Africa
Users	42.9	25.7	10.6	10.2	6.4	3.6	0.6
Contributors	43.0	27.4	10.5	9.0	5.5	3.9	0.6
Contributions	47.5	27.3	8.4	8.6	3.6	4.1	0.5
Receivers	44.3	26.6	9.9	9.2	5.4	4.0	0.6
Received contributions	50.9	26.5	7.4	7.3	3.2	3.9	0.7
Users with watched repositories	42.6	25.9	10.4	10.5	6.2	3.7	0.6
Being watched	58.9	20.8	6.1	5.4	3.7	4.6	0.4
Followed users	44.5	25.7	9.9	9.5	6.3	3.6	0.5
Being followed	55.4	20.4	5.8	9.3	5.0	3.7	0.3

Table 3: Github participation by region (%).

3.6% of the contributions. (A large part of this loss comes from Brazil, which accounts for 4.2% of users and 2.2% of contributions.)

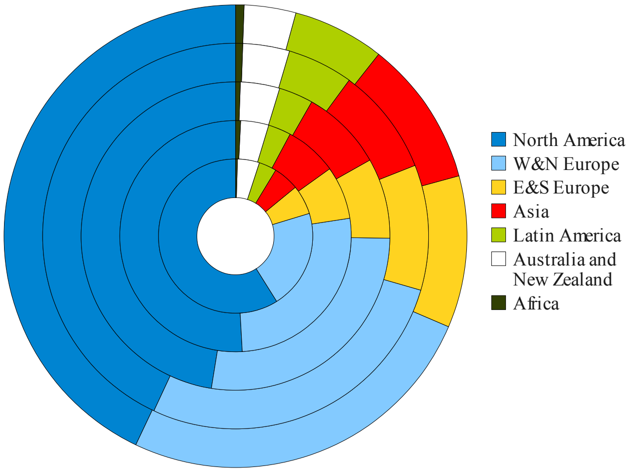
Despite the dominance of North America and W&N Europe, it is important to note that other regions do jointly account for 31% users and 30% of the contributions.

Users and Contributions by Local Cluster

Beyond macro-regional differences, users and their contributions are concentrated in a handful of locations. The top five local clusters – "San Francisco," "London," "New York," "Tokyo" and "Boston" – account for 20% of registered users and 25% of the contributions. The top ten, shown in table 3, account for 29% and 35% of users and contributions respectively. The top fifty account for 63% and 71%. Looking at the clusters, at the same time, makes it easier to note the presence of users outside North America and W&N Europe: the fourth largest cluster is Tokyo, while Brazil's São Paulo ranks 12th by he number of users and 14th by he number of contributions.

Distance and Ties

The Github dataset, however, allows us to not only look at where the users are, but also at how distance affects ties between them. We identified 51,507 contributor-owner pairs – cases of a registered user making a contribution to an unforked repository of some other user. 33% of those pairs could be located on *both* end. Of those, 24% fall within the same local cluster. Contributor-owner pairs, however, vary in the number of contributions made. As it turns out, contributor-owner pairs located in the same cluster have more contributions on average. If we add up all contributions, 41% of the total begin and end in the same cluster.



From outer ring to inner:
share of users,
share of contributors,
share of contributions,
share of received contributions,
share of watched repositories

Figure 1: Five participation metrics as a diagram.

It appears that one of the reasons for the locality of contributions is the fact that contributors and repository "owners" may be associated with the same organizations.

Cluster	Share of users	Share of contributors	Share of contributions
San Francisco, USA	7.4	7.4	9.7
London, UK	4.2	4.4	3.7
New York, USA	3.9	3.7	4.1
Tokyo, Japan	2.6	2.7	3.2
Boston, USA	2.2	2.4	3.3
Seattle, USA	2.0	2.0	2.2
Chicago, USA	1.9	2.1	2.4
Washington, USA	1.8	2.1	1.9
Los Angeles, USA	1.8	1.6	2.9
Paris, France	1.6	1.8	1.6

Table 4: Github participation by local clusters (%).

Users have an option to specify their "company" in their profiles, and we checked whether the contributor's and the owner's reported company matched. In-company contributions accounted for 22% of all contributions and 74% of such contributions were within the same local cluster. Even cross-company contributions, however, tended to be local: 32% were in the same cluster.

There was also a substantial – though weaker – tendency towards locality for user-following: 30% of following ties are local. This tendency is still observable – but yet lower – for repository-watching: only 8% of such ties are local. (The number rises slightly – to 11% – if we count separately each watched repository, allowing for multiple watching ties between the same pair of users. As with contributions, local pairs users watch a larger number of repositories.) The share of local repository watching pairs goes down to 7% if we consider only cross-company ties.

We interpret the difference between repository-watching and following as reflecting a difference between instrumental and social ties. Users likely "watch" repositories for software their use in one way or another. They "follow" people whom they find interesting as individuals. The latter relation is much more likely to reflect pre-existing social ties, often local ones.

The high prevalence of local ties can be explained in part by the high degree of local clustering noted earlier. When users are concentrated in a handful of places, they will form a large number of local ties even if ties are formed fully at random. The observed prevalence of local ties, however, far surpasses this clustering effect. For example, if users in the San Francisco cluster made contributions choosing recients at random, about 7% of their contributions would land back in the San Francisco cluster. For users located in other local clusters, the number would be much smaller, due to those clusters' smaller share of the total user population. Overall, at the current level of clustering, we would expect about 2% of the ties to fall within clusters if the ties were formed randomly.

Distance also matters for longer ties. Figure 2 shows a histogram of contribution and following ties grouped by length into 300 km buckets. The three-hump shape of the histogram is explained by the fact that the users are distributed unevenly around the globe. Since a large number of the users are concentrated on the two coasts of the United States and in Western Europe, we we can expect a substantial number of contributions with the length of a little over 4,000 km (the distance between the two coasts) and around 8,000-9,000 km (the distance from California to Western Europe). On the other hand, we can expect relatively few contribution ties with length of 5,000 km, since there are no major clusters located at that distance from such major centers as San Francisco, New York or London. (A circle 5,000 km in radius centered on San Francisco would be located almost entirely in the ocean or the Canadian North.) The histogram includes a line that shows the percentage of ties that we would expect to fall in each bucket if the users formed connections randomly while remaining in their current location.[9] (This line was obtained through a simulation.) As we can see, the general shape of this simulated distribution repeats the shape of the observed distributions, but there is a substantial deficit of connections past 5,000 km, which is easiest to see in the range from 7,000 to 10,000 km.

Asymmetries

While users in North America make a disproportionate number of contributions (relative to the total number of registered users), they account for a yet larger share of contributions received 51% vs 48%. Repositories associated with North American accounts also account for 59% of repository-watching. For most other regions, the opposite is true. E&S Europe, Asia and Latin America, jointly account for 21% of contributions made, but only 18% of contributions received. (See table 3.) They account for a yet smaller share of repositories being watched: 16%. North American users are also being followed at a disproportionate rate. Though this effect is substantially weaker than for repository-watching, this weakening appears to be in part due to the stronger locality of following.[10]

[9] Note that if the actual distribution were not taken into consideration – that is, if the users were placed randomly on the map and then formed random connections – we would expect the number of ties to grow steadily up to 10,000 km, then reduce back to nothing at 20,000 km, reflecting the shape and dimensions of the globe.

[10] We do not present here an analysis of ties between local clusters, because such analysis is complicated by fact that North America accounts for a larger number of clusters and thus scores disproportionately on all cross-local ties.

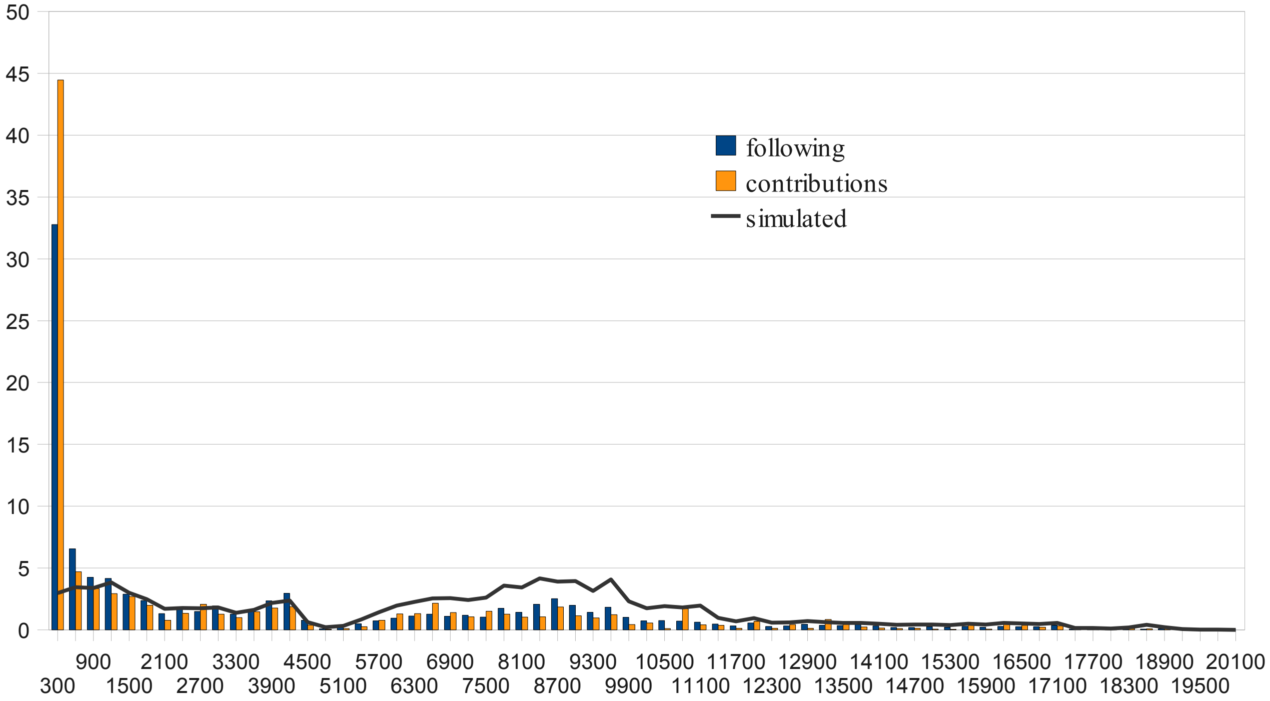


Figure 2: Distribution of ties by length, using 300 km buckets.

CONCLUSION

Our analysis of the geographic dimensions of open source participation on Github provides a picture of a distributed yet clustered world. Participants are spread around the world, though concentrated substantially in some regions, generally replicating the proportions found by other researchers. However, they are closely clustered in a relatively small number of places, lending some evidence to Carmel and Agarwel's prediction of a global virtual archipelago of groups of cooperating developers (2001). Distance has an important effect on code contribution and following of users, with a lesser effect on the attention directed towards software repositories. Place matters in open source software development.

ACKNOWLEDGMENTS

This material is based in part on work supported by an award from Project Open Source|Open Access, KMDI.

REFERENCES

1. Barcellini, F., Détienne, F., Burkhardt, J., and Sack, W. A socio-cognitive analysis of online design discussions in an Open Source Software community. *Interacting with Computers 20*, 1 (2008). 141-165

2. Bird, C., Rigby, P. C., Barr, E. T., Hamilton, D. J., German, D. M., and Devanbu, P. The promises and perils of mining git. In Proc. *MSR '09.* IEEE Computer Society (2009).

3. Bradner, E. and Mark, G. Why distance matters: Effects on cooperation, persuasion and deception. In *Proc. CSCW 2002*, ACM Press (2002). 226-235.

4. Carmel, E., and Agarwel, R. Tactical Approaches for Alleviating Distance in Global Software Development. *IEEE Software 18*, 2 (2001). 22-29.

5. Crandall, D., Backstrom, L., Huttenlocher, D., and Klienberg, J. Mapping the world's photos. In *Proc. WWW 2009*. ACM Press (2009).

6. Crowston, K. and Howison, J. The social structure of open source software development teams. *First Monday 10*, 2 (2005). Online.

7. Crowston, K., Wei, K., Li, Q., and Howison, J. Core and periphery in Free/Libre and Open Source software team communications. In *Proc. HICSS '06*. IEEE Computer Society (2006).

8. Del Rosso, C. Comprehend and analyze knowledge networks to improve software evolution. *J. Softw. Maint. Evol.: Res. Pract. 21* (2009). 189-125.

9. Demazière, D., Horn, F. and Zune, M. Individual and organizational approaches of voluntary participation in FLOSS: An analysis based on the case of Spip. In *Proc. 1st Workshop on the diffusion of FLOSS and the Organisation of the Software Industry: From Social Networks to Economic and Legal Models.* (2007).

10. Dempsey, B. J., Weiss, D., Jones, P. and Grenberg, J. A

quantitative profile of a community of open source Linux developers. *Unpublished working paper*, School of Information and Library Science, University of North Carolina at Chapel Hill (1999). http://sils.unc.edu/research/publications/reports/TR-1999-05.pdf

11. Dinh-Trong, T. T. and Bieman, J. M. The FreeBSD project: A replication case study of open source development. *IEEE Transactions on Software Engineering 31*, 6 (2005). 481-494.

12. Ducheneaut, N. Socialization in an Open Source Software Community: A Socio-Technical Analysis. In *Proc. CSCW 2005*, ACM Press (2005). 323–368.

13. German, D.M. The GNOME project: A case study of open source, global software development. *Software Process Improvement and Practice 8* (2003). 201-215.

14. Ghosh, R. A., Glott, R. Krieger, B. and Robles, G.. Free/ libre and open source software: Survey and study. *Report, International Institute of Infonomics*, University of Maastricht, Maastricht, The Netherlands, 2005.

15. Ghosh, R. Understanding free software developers: Findings from the FLOSS study. In Feller, J., Fitzgerald, B., Hissam, S. and Lakhani, K.R. (eds.), *Perspectives on Free and Open Source Software.* Boston, MA: MIT Press, Boston, MA, USA, 2005.

16. Gonzalez-Barahona, J.M, Robles, G., Andradas-Izquierdo, R., and Ghosh, R.A. Geographic origin of libre software developers. *Information Economics and Policy* 20 (2008). 356–363.

17. Gutwin, C., Penner, R., Schneider, K. Group Awareness in Distributed Software Development. In *Proc. CSCW 2004.* ACM Press (2004). 72-81.

18. Guy, I., Jacovi, M., Perer, A., Ronen, I. and Uziel, E. Same places, same things, same people? Mining user similarity on social media. In *Proc. CSCW 2010,* ACM Press (2010). 41-50.

19. Hecht, B. and Gergle, D. On the "localness" of user-generated content. In *Proc. CSCW 2010,* ACM Press (2010). 229-232.

20. Herbsleb, J.D., Mockus, A., Finholt, T.A., and Grinter, R.E. An empirical study of global software development: Distance and speed. In *Proc. ISCE 2001*. IEEE (2001). 81-90.

21. Holman, Z. One million repositories. *GitHub Blog* (2010). http://github.com/blog/685-one-million-repositories.

22. Holmström, H., Fitzgerald, B., Ågerfalk, P.J., and Conchúir, E.O. Agile practices reduce distance in global software development. *Information Systems Management 23,* 3 2006. 7-18.

23. Kuk, G. Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. *Management Science 52*, 7 (2006). 1031-1042.

24. Lave, J. and Wenger, E. *Situated Learning: Legitimate Peripheral Participation.* Cambridge: Cambridge University Press, 1991.

25. Lundell, B., Lings, B., Ågerfalk, P.J., Fitzgerald, B. (2006). The distributed open source software development model: Observations on communication, coordination and control. In *Proc. ECIS 2006,* (2006).

26. Marshall, A. Book IV: The Agents of Production. Chapter X. Industrial organization, continued. The concentration of specialized industries in particular localities" In *Principles of Economics: An Introductory Volume,* London: McMillan and Co., 1890/1927. 267–277.

27. Mockus, A., Fielding, R. T., Herbsleb, J. D. Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology 11*, 3 (2002). 309-346.

28. Olson, G.M. and Olson, J.S. Distance matters. *Human-Computer Interaction 15* (2000). 139-178.

29. Robles, G. and Gonzalez-Barahona, J.M. (2006). Geographic Location of Developers at SourceForge. In *Proc. MSR'06*, Shanghai, China.

30. Short, J. A., Williams, E., and Christie, B. *The social psychology of telecommunications.* John Wiley & Sons, New York, 1976.

31. Spinellis, D. Global software development in the freeBSD project. In *Proc. GSD '06*, ACM Press (2006). 73-79.

32. 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 (2008). 173–186.

33. Tang, R., Hassan, A. E. and Zou, Y. A Case Study on the Impact of Global Participation on Mailing Lists Communications of Open Source Projects. In *Proc. KCSD 2009,* JSAI (2009). 63-76.

34. Torvalds, L. Linus Torvalds on git. Google Tech Talks [video]. Available at http://www.youtube.com/watch?v=4XpnKHJAok8

35. Tuomi, I. Evolution of the Linux credits file: Methodological challenges and reference data for open source research. *First Monday 9*, 6 (2004). Online

36. Von Engelhardt, S., Freytag, A. and Schulz, C. On the Geographic Allocation of Open Source Software Activities. *Jena Economic Research Papers*, (2010).