

Visualizing Time and Geography of Open Source Software with Storygraph

Ayush Shrestha

Department of Computer Science
Georgia State University
Email: ashrestha2@cs.gsu.edu

Ying Zhu

Department of Computer Science
Georgia State University
Email: yzhu@cs.gsu.edu

Ben Miller

Department of Communication
Georgia State University
Email:miller@gsu.edu

Abstract—Free/Libre and Open source software are generally maintained by a group of developers contributing to the software voluntarily without the presence of any governing institution. Online collaboration platforms and sub-version systems allow developers from all over the world with the necessary skills to contribute to the software. Recently, there has been much interest in analyzing the developers' geographic location since it also serves as a socio-economic marker. In this paper we present our spatio-temporal visualization technique called Storygraph that shows the developers, developer locations and the frequency of commits based on the commit log in one integrated view. We also apply our techniques to the VCS of Rails, Homebrew and D3.js obtained from GitHub.

I. INTRODUCTION

Free/Libre and Open Source Software (FLOSS) are in most cases created and maintained by developers voluntarily. Since the collaboration between developers takes place online without any governing institutions, detailed designs, project plan or list of deliverables, developers globally can contribute to the software[1]. This is further facilitated by online open source repositories like GitHub, SourceForge, Google code, CodePlex, etc. Even though open source software projects may seem to involve developers from all around the world, literature show a strong geographic bias of these developers towards North America and Europe [2][3][4][5].

FLOSS provide good platforms for studying distributed software development since most of them are global efforts. Prior work has studied how the difference in geographical locations and time affect the nature of collaboration and the quality of the product. However, most of these analyses did not use data visualization, even though visualization is a valuable technique for analyzing geographical information. Heller, et al. [6] and Xu, et al. [7] developed methods to visualize geographic locations and the collaboration or relationships among the developers. However, these methods do not include time information. Our work improves on the previous work by integrating geographic and temporal information in one visualization.

In this paper we aim to visualize the geographic location of the developers in FLOSS projects together with their time of commits. Presenting time and location on a single chart is challenging. One way to represent both on the same chart is using 3D visualizations. However, 3D visualizations suffer from glyph occlusion and cluttering when the size of the log

increases. To address these issues, we apply our previous 2D spatio-temporal visualization technique, *Storygraph* [8], on the commit logs of three open source software from GitHub.

II. CONCEPT

The Storygraph is composed of two parallel vertical axes V_{lat} and V_{long} and an orthogonal horizontal axis H . As in Cartesian coordinates, the values in the axes are ordered in ascending order from top to bottom in the vertical axes and from right to left in the horizontal axis. The vertical axes represent latitude and longitude of a location while the horizontal axis represents time. In order to plot a point on a Storygraph, a precise location and a time stamp is required. Given a commit time stamp together with the location of the developer, a line segment connecting the latitude and longitude on the vertical axes corresponding to the developer's location is drawn in the Storygraph. A marker is then placed on this line at the time of the commit. Multiple commits from the same developer are represented as multiple markers along that location line. We assume that the location of the developers are fixed in this paper.

An example of Storygraph is shown in Figure 1. The top figure shows code commits from developers in three locations (x_1, y_1) , (x_2, y_2) and (x_3, y_3) at different times $t_1 - t_5$. The bottom figure shows the same coordinates plotted on the Storygraph. It can be observed that the temporal aspect is more evident in the Storygraph as compared to the traditional map.

A. Mathematical model

Given a code commit at time t by a developer in a location with geo-coordinates (α, β) , first a line segment connecting latitude and longitude on the two vertical axes is drawn. This line is called a *location line* and represents real geographic location. Second, a marker is placed along this location line at time t .

The function $f(\alpha, \beta, t) \rightarrow (x, y)$ which maps an event to the 2D Storygraph plane can be formally written as follows:

$$y = \frac{(\beta - \alpha)(x - T_{min})}{T_{max} - T_{min}} + \alpha \quad (1)$$

$$x = t \quad (2)$$

where T_{min} and T_{max} are the maximum and minimum time stamps within the data set.

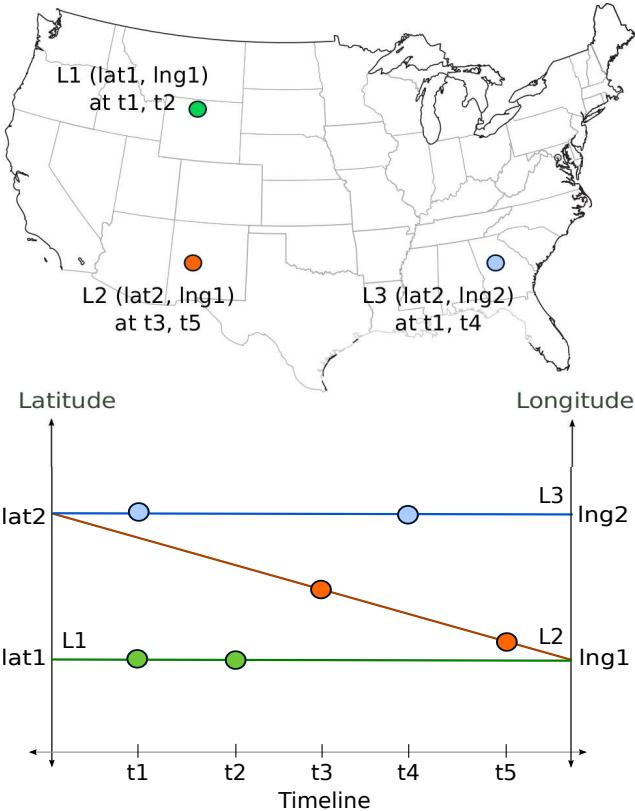


Fig. 1. Top: A geographic map showing commits from three locations $L_1 - L_3$ at different times $t_1 - t_5$. L_1 and L_2 share the same longitude, l_{ng2} and L_2 and L_3 share the same latitude, l_{at2} . Bottom: Same set of events plotted on the Storygraph.

B. Interpretation

One of the important properties of Storygraph is its ability to show code commits from multiple locations and times on a single chart. This helps to show where the developers are most active and also when the developers are most active during the software life cycle.

1) *Different locations, same time:* Multiple code commits at a unit time are represented as a vertical cluster of points at that time. An example of many code commits at one time can be observed in Figure 1 (bottom) at t_2 and the areas pointed by B and C in Figure 2.

2) *Same location, different time:* Multiple code commits from a certain location are represented as a cluster of points along the location line eg. Figure 1(bottom) - all three location lines have two events each. This can also be observed in Figure 3 in location lines marked by *US* and *UK*.

C. Location lines

Location lines are a crucial part of Storygraph. Without location lines, a point marker on the graph could belong to multiple locations. By geometry, the location lines of two proximal locations on a map are also close on a Storygraph. Despite this, in situations where data is dense enough to reveal meaningful patterns, drawing location lines may lead to over plotting. In such cases, the location lines may be turned off.

Storygraph without location lines only shows the number of commits per unit time.

To prevent cluttering resulting from the location lines, we paint the lines with varying intensity of the same color as shown in Figure 2 and Figure 3. The intensities in this case directly correspond to the frequency of commits from a location - higher the number of commits, darker the location line. Thus given the maximum and minimum number of commits for a repository, n_{max} and n_{min} , and a tuning parameter p , the color c of the location line is given by:

$$c = \left(\frac{n - n_{min}}{n_{max} - n_{min}} * 255^{(1/p)} \right)^p \quad (3)$$

where n is the number of commits for the current line and p is the tuning parameter.

Increasing the values of p highlights the outliers (locations which have large number of commits). However, Equation 3 requires the location data to be noiseless when using large values for p .

III. IMPLEMENTATION

Our implementation of the Storygraph consisted of a three stage pipeline.

Stage I: Getting the log and the location of the developers. For this purpose, we chose GitHub.com since it also had developer profiles with location. We used git to get the list of the commit logs from the server. Obtaining the names of the contributors from the log, we wrote a script to query GitHub for profiles of each user. We used an XML parser to extract the location of the users from the resulting pages. One of the challenges that we faced in this stage was when developers did not have an address. In these cases, we queried the LinkedIn.com database and personal websites if listed.

Stage II: Removing noise and using Google Map API for geocoding. This stage was automated using script which called the geocoding function in Google Map API. The challenges we faced in this stage included multiple locations and vague addresses. Many developers had more than one location listed in their profiles e.g. "London, Tokyo, San Francisco", "Atlanta, New York" etc. Having more than one location resulted in more than one pair of latitude and longitude. For simplicity we only considered the first location and discarded the remaining locations to address this issue. Developers who had vague addresses e.g. "Somewhere in Earth", "Dark side of the moon" etc, were either looked up manually or discarded based on their number of commits.

Stage III: Setting up database and visualizing the Storygraph. We implemented the Storygraph in Microsoft .NET using WPF Framework. The data was stored in MySQL database.

IV. APPLICATION

We plotted the commit history of three projects from GitHub: *Rails*, *D3.js* and *Homebrew* on the Storygraph.

Figure 2 shows the Storygraph of *Rails*. Also known as Ruby on Rails, *Rails* is a framework running on Ruby

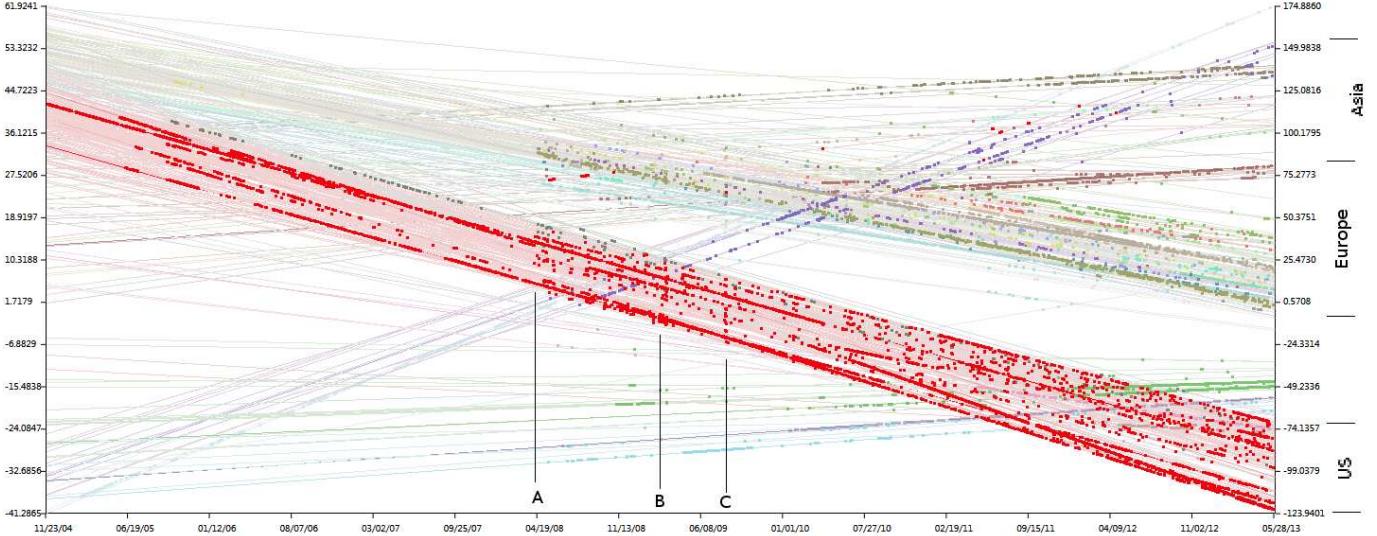


Fig. 2. Storygraph of *Rails* development based on the GitHub VCS. Up until time marked by *A*, there were only few developers, majority of them in United States. After *A*, there were many commits from all over. *B* and *C* show many code commits within a short span of time from many locations. *US*, *Europe* and *Asia* show the bands formed due to the clustering of location lines. Location lines have been painted with $p = 0.8$ to subdue locations with smaller number of commits.

programming language. Rails facilitates the communication between pages/applications and servers. The first version of Rails was released in December 13, 2005 and is still active. From the figure, we can see that until the time marked by *A*, most of the code commits were centralized in the US with few being in Europe and Asia. After time *A*, there were many code commits from developers all over the world. Markers *B* and *C* show multiple code commits within a short span of time from many different locations. This is generally the case before or after a major version release.

The Storygraph of D3.js is shown in Figure 3. D3, standing for Data-Driven Documents is a W3C compliant Javascript visualization library. The first version of D3 was released in 2011 and remains active. Unlike *Rails* or *Homebrew*, two dominating contributors can be seen in the figure marked by *US* and *UK*. The set of commits marked by *US* corresponds to Michael (Mike) Bostock, the founder of the project from United States and *UK* corresponds to Jason Davies from United Kingdom.

Figure 4 shows the Storygraph of the OS X package manager *Homebrew*. Even though Homebrew has yet to release version 1.0 yet, it has been able to attract a lot of developers in US, Europe and Asia. Similar to *Rails*, it can be seen from the figure that after a certain period of time marked by *A*, a lot of people started contributing to the code from different locations.

Few inferences can be made from Figures 1-3:

- There is a time period in the start of open source projects where there is only a sole developer working on the project. After that point, the contributors tend to increase rapidly.
- In most cases, the developers contributing at the start of the project continue to contribute to the code actively till

the end. This trait is more clearly observed in Figure 3.

V. RELATED WORK

Some previous works [9][10] have tried to create storylines for open source projects. But they focus on timelines without considering geographical locations of the developers. However, some works [4][2][11] have shown that geographical location is important for analyzing the nature of collaborations and the effectiveness of distributed software development. These works, however, have not used visualization for their geographical location analysis. Heller, et al. [6] proposed three techniques (map with links, small multiples, and matrix diagrams) to visualize the geographical locations of the developers as well as their collaborations. However, their methods do not have integrated time information. Xu, et al. [7] proposed a graph chart to visualize the relationships among developers from different regions. That method did not integrate temporal information.

VI. DISCUSSION AND CONCLUSION

We presented Storygraph and applied our technique to visualize the time of the commits and the location of developers on a single chart. Our visualization shows a narrative of how the developers all over the world collaborated to create FLOSS based on the data from GitHub. The main advantage of Storygraph over maps or timelines is that it can show the temporal aspect of the data without losing the big picture. However, it does not consider the user contribution when highlighting the location lines and also does not show where and which files were modified. We intend to incorporate these in our future design. We also intend to create a web version and collect user feedback to evaluate the effectiveness of our visualization.

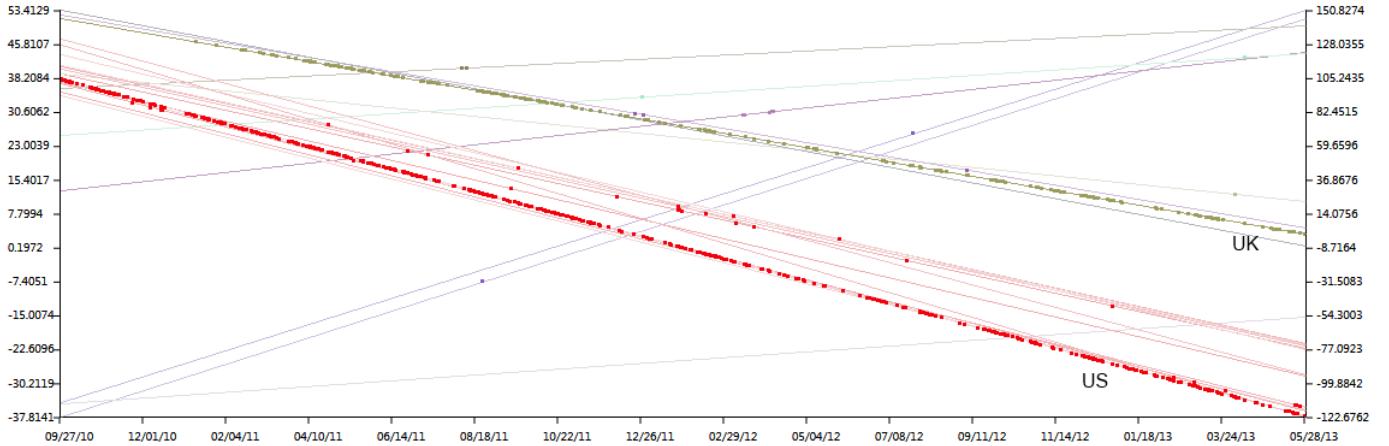


Fig. 3. Storygraph of *D3.js* development based on the GitHub VCS. It can be observed that developers from locations marked by *US* and *UK* have the highest number of contributions. The location lines have been painted with $p = 0.18$ to include locations with smaller number of commits.

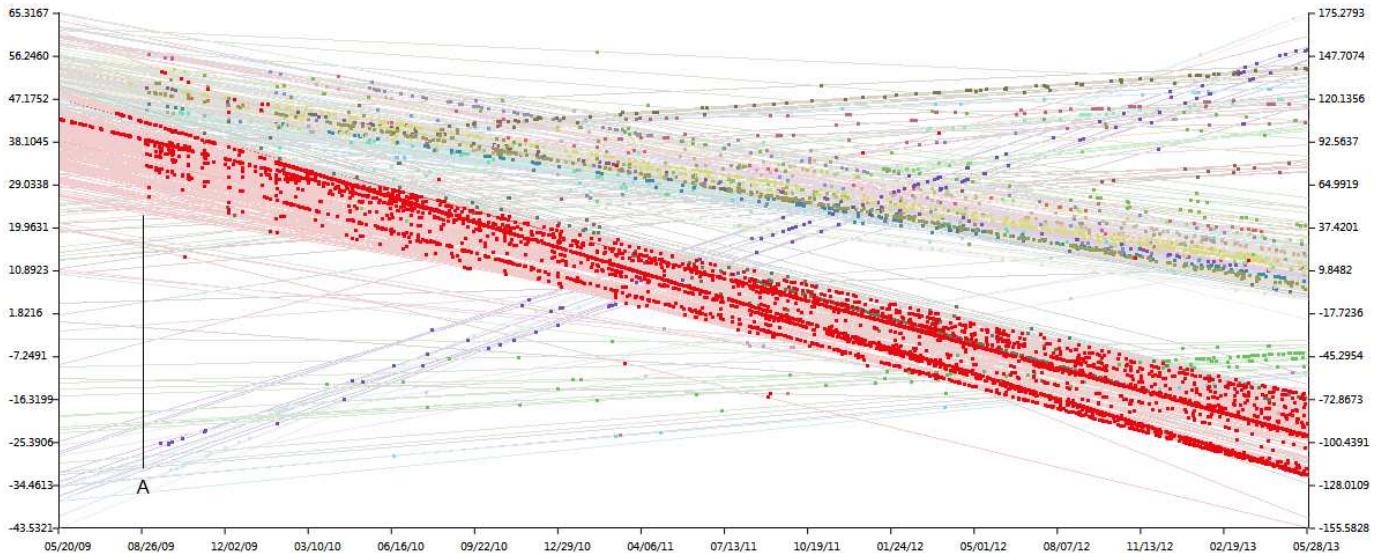


Fig. 4. Storygraph of *Homebrew* development based on the GitHub VCS. The number of developers starts to increase after time marked by *A*.

REFERENCES

- [1] S. Weber, *The success of open source*. Cambridge Univ Press, 2004, vol. 368.
- [2] J. M. Gonzalez-Barahona, G. Robles, R. Andrade-Izquierdo, and R. A. Ghosh, “Geographic origin of libre software developers,” *Information Economics and Policy*, vol. 20, no. 4, pp. 356–363, 2008.
- [3] K. Crowston, K. Wei, J. Howison, and A. Wiggins, “Free/libre open-source software development: What we know and what we do not know,” *ACM Computing Surveys (CSUR)*, vol. 44, no. 2, p. 7, 2012.
- [4] Y. Takhteyev and A. Hiltz, “Investigating the geography of open source software through github,” 2010. [Online]. Available: <http://www.takhteyev.org/>
- [5] G. Robles and J. M. Gonzalez-Barahona, “Geographic location of developers at sourceforge,” in *Proceedings of the 2006 international workshop on Mining software repositories*. ACM, 2006, pp. 144–150.
- [6] B. Heller, E. Marschner, E. Rosenfeld, and J. Heer, “Visualizing collaboration and influence in the open-source software community,” in *Proceedings of the 8th Working Conference on Mining Software Repositories*, ser. MSR ’11. New York, NY, USA: ACM, 2011, pp. 223–226.
- [7] X. Ben, S. Beijun, and Y. Weicheng, “Mining developer contribution in open source software using visualization techniques,” in *2013 Third International Conference on Intelligent System Design and Engineering Applications (ISDEA)*, 2013, pp. 934–937.
- [8] A. Shrestha, Y. Zhu, B. Miller, and Y. Zhao, “Storygraph: Telling stories from spatio-temporal data,” in *Lecture Notes in Computer Science*, vol. 8034. Springer, 2013, pp. 693–703.
- [9] A. Kuhn, D. Erni, P. Loretan, and O. Nierstrasz, “Software cartography: Thematic software visualization with consistent layout,” *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 22, no. 3, pp. 191–210, 2010.
- [10] M. Ogawa and K.-L. Ma, “Software evolution storylines,” in *Proceedings of the 5th international symposium on Software visualization*. ACM, 2010, pp. 35–42.
- [11] A. Sarma and A. Hoek, “Palantir: Increasing awareness in distributed software development,” in *Proceedings of 25th International Conference on Software Engineering (ICSE)*, 2003, pp. 444–454.