Mining Developer Contribution in Open Source Software Using Visualization Techniques

Xu Ben, Shen Beijun,Yang Weicheng

School of Software Engineering, Shanghai JiaoTong University, Shanghai, 200240, China
shaci1234@yahoo.com.cn

***Abstract*—the research of developers' contribution is an important part of the software evolution area. It allows project owners to find potential long-term contributors earlier and helps the newcomers to improve their behaviors. In this paper, we examined the contribution characteristics of developers in open source environment based on visual analysis, and presented approaches from three aspects-influencing factors, time characteristics and region characteristics. Our analysis used data from github and revealed some regular patterns. We found that the code which newcomers started to contribute with more people engaged in would lead to less contribution in some degree. We also found that there's a relation between developers' early and later period contribution. In addition, developers from different regions were more likely to have dominant relationship. Our findings may provide some support for future research in the area of software evolution.**

Keywords- open source software; software evolution; visual analysis; contribution characteristics

I. Introduction

The study of software evolution has experienced a shift from commercial software to open source software. From 1972 to 2002, Lehman and his team proposed eight laws of software evolution according to the research of commercial software [1]. But recent research demonstrates that these laws are not complied with especially in open source software environment [2]. Some researchers tried to improve the theoretical models of software evolution through new perspectives. For example, Jiangwei Wu had applied self-organized criticality to software evolution [3].

Apart from that, more efforts are put into the research of software quality, productivity and developer contribution. Among these three, the former two have overwhelming advantages in amount over the latter one; meanwhile, developer contribution research mainly focuses on the measure of contribution.

In this paper, we attempt to combine the techniques of data mining and visual analysis, investigating the developer contribution characteristics from brand new perspectives. The research questions and key contributions of this paper are the following:

RQ1. What factors with respect to code will impact developers' contribution?

Parallel coordinates is employed in this study. We find that the code which newcomers started to contribute with more people engaged in will lead to less contribution in some degree.

RQ2. Is there any pattern in the time distribution of contributors' commits?

Contribution is divided into early and later period according to the longest no-commit interval. We find that there's a relation between developer's early and later period contribution.

RQ3. Is there any relation between developers' contribution and their region?

Dominant diagram is employed in this study with region information added. We find that developers from different regions are more likely to have dominant relationship.

Our findings on one hand allow project owners to find potential long-term contributors earlier and pay more attention to them, on the other hand help the newcomers to improve their behaviors and increase their contribution to the project.

II. Related Work

In the past few decades, many researchers had conducted related researches.

Minghui Zhou, Peking University, proposed to model the individual's chances to become a valuable contributor through her capacity, willingness, and the opportunity to contribute at the time of joining [4]. Using issue tracking data of Mozilla and Gnome, they find that the probability for a newcomer to become a long term contributor is associated with her initial contribution type, bug report style, whether succeed to get at least one reported issue to be fixed, peer group and so on.

[5], [6], [7] studied the effect of geographic distribution on software productivity and quality using regression analysis method. In [7] the cooperation relations between the developers were also involved and the distance between them was actually calculated. Interestingly, [6] and [7] came up with opposite conclusions in the analysis of geographic impact to software quality.

Evolution Matrix [8], Evolution Spectrographs [9], Timeline [10] are visualizations based on matrix with x axis represents time and y axis consists of software component, different metrics can be applied on item value.

Icicle plot is a variant of tree with more compact structure, so multiple versions can be displayed in the 2-D plane. It's employed in Code Flow [11], Hierarchical Edge Bundles [12] to visualize the evolution of code structure.

In addition there is city-based visualization [10] [13], as well as using animation [14].

In the subject of developer activities evolution, Ogawa put forward varieties of visualization techniques, including

Storylines [15], Code Swarm [16] and StarGate [17]. Code structure is also concerned in StarGate, but it's static without taking evolution into consideration.

All researches mentioned above except Minghui Zhou's do not focus on developers' contribution characteristics. While Minghui Zhou's study mainly uses regression analysis, in this paper, we explore from three new perspectives with the help of visual analysis.

III. METHODOLOGY

A. *contribution measure*

Traditionally, the most classic and simple measure of contribution is line of code (LOC). Gousios et al proposed a more complex system concerning both code and non-code contribution [18]. In our study, we only considered LOC and number of commits. The i-th developer's contribution is calculated as follows:

$$Contri(i) = \frac{LOC(i)}{\sum_{k=1}^{n} LOC(k)} + \frac{Commit(i)}{\sum_{k=1}^{n} Commit(k)}, 1 \le i \le n$$

n is the total number of developers.

B. *study of influencing factors*

We examined the influencing factors of developers' contribution from the code they started to commit at first time. We studied code instability, code structure as well as commit features. Instability is measured using the number of modifies by others one month before the newcomer's first commit (HCC). FanIn, fanOut, LOC, number of attributes (NOA) and number of methods (NOM) are used to measure code structure. Commit features include LOC changed (FCLOC) and the number of new files (NFC) in the first commit.

We can easily get HCC、FCLOC and NFC from commits history. As to code structure, we exported codes before and after developer's initial commit through git command followed by parsing them with Moose tools. Moose is a well-known static analysis tool from which we can obtain metrics required directly.

We combined regression analysis with visual analysis methods. In regression analysis, we used the Matlab tools to conduct univariate and multivariate linear regression. And we used parallel coordinates in visual analysis. Parallel coordinates is a common way of visualizing high-dimensional geometry.

C. *study of time characteristics*

In this section, we created barcode visualization. It consists of several vertical parallel line segments of equal length. These segments are sequentially arranged from left to right in accordance with time. A line segment represents one or more commits on a certain day. The relative amount is indicated by the color of the line with deeper color means more commits. The gaps between two line segments are the periods with no commits.

In this paper, barcode visualization was used to study developer's inert period. Inert period is defined as the maximum gap of developer's commit history. We visualized every developer's commit history as a barcode graph and then put the entire project's barcode below to observe the consistency of the two. We also divided developer's contribution into early stage and later stage according to the inert period and fatherly studied the relationship between them.

We used regression analysis to mathematically verify the results. We calculated the length of the inert period and others' number of commits during this time followed by making linear regression with developer contribution respectively. Meanwhile, we also used regression analysis to find out the relationship between early and later stage contribution.

D. *study of region characteristics*

Dominant diagram was used in the study of region characteristics. It's a simple directed graph with vertex represents contributor and edge represents dominant relation on certain files, the arrow points to the dominator. A file's dominator is the one contributes most to that file. When a developer modifies a file, dominant relation arises between him and the dominator. The dominator will change over time, but the dominant relationship remains until developer re-contribute to the file. We made an improvement by using color to represent the contributor's region and visualized two dominant diagrams according to whether contributor and dominator come from the same region.

For various reasons, most developers did not provide their region information directly, so we used a variety of techniques to predict. We referred to the methods mentioned in [6], analyzing through email domain names, time zone, company info and searched on web sites as well as Google, LinkedIn, Twitter. We used nation as the unit of region, because on the one hand, nation info is much easier to gather than the specific urban info; on the other hand, nation reflects not only space distance, but also language and culture differences.

IV. CASE STUDY

A. *Dataset*

We collected data from CraftBukkit, an open source project hosted on github. As the extensive application of Git version control system, more and more projects are ported to github. CraftBukkit is an API implementation of the Minecraft Server Mod developed using Java language. It's a relative young project started since December, 2010. We collected its commit and developer information generated from December 2010 to July 2012.

B. *results of influencing factors*

The regression results between multiple metrics of code dimension and the contribution are not satisfactory (Table 1). As it can be seen from the table, the maximum R-squared occurs in the relation between the number of new files (NFC) and the contribution, which is about 0.3.

Table 1 The univariate linear regression results

Metrics	Linear coefficient b	R-squared
HCC	-0.0005	0.0115

FCLOC	0.00004	0.2649
NFC	0.0123	0.2999
fanIn	-0.0003	0.0226
fanOut	-0.0003	0.0319
LOC	-0.00003	0.0325
NOA	-0.0003	0.0104
NOM	-0.0002	0.0297

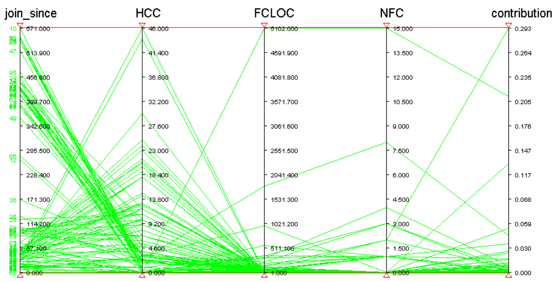

Figure 1. parallel coordinates of metrics on code dimension

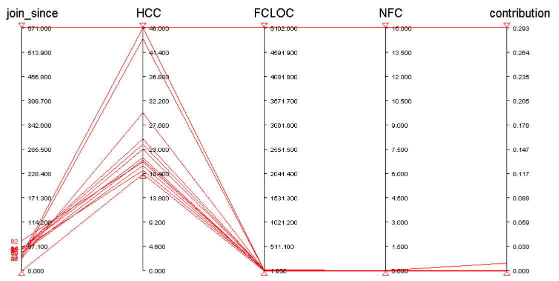

Figure 2. parallel coordinates with lower limit of HCC

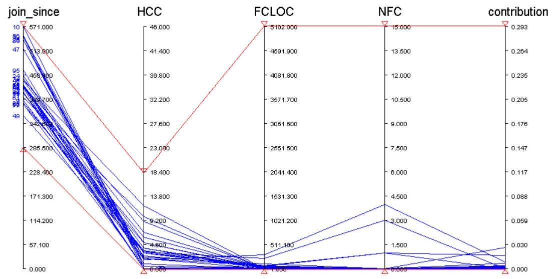

Figure 3. parallel coordinates with lower limit of join_since

We applied parallel coordinate visualization in the same data (Figure 1). In figure 1, join_since represents days of developer's join time since the project's first commit. By adjusting the lower limit of HCC, we found that developers with higher HCC had lower contribution (Figure 2). On the contrary, it means that developers with higher contribution appeared in the part of lower HCC. Because HCC measures the number of modifies one month before first commit instead of the number of modifies since its creation, the interference of join time is avoided to some extent. Of course, for a more thorough description of the problem, we adjusted the lower limit of join_since among developers with lower HCC (Figure 3). The results showed that they still hold higher level of contribution than the higher-HCC group. Usually we think that the number of modifies stands for the instability of the code. Unstable code will give developers more opportunities to contribute. But on the other hand, with more people involve in, the average contribution for individuals will decrease.

Law 1: developer contribution is influenced by the instability of the code he or she starts to contribute. The code with more developers involve in will lead to less contribution in some degree.

C. *results of time characteristics*

The results of barcode visualization are shown in Figure 4. We only intercepted part as the whole diagram is too large. It seemed that the consistency of developer's barcode and project's barcode may not affect the contribution of the developer. The further linear regression analysis confirmed our thoughts. In addition, it can be found that most developers' early stage and later stage contribution had no much difference, and the R-squared of linear regression surprisingly reached 0.7488. We believe it's a discovery with great significance because we can predict future contribution simply by finding the inert period so far.

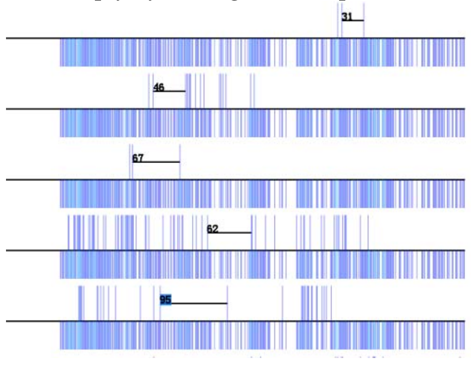

Figure 4. part of the barcode visualization

Law 2: linear relationship exists between developers' early and later stage of contribution.

D. *result of region characteristics*

We analyzed the region composition of CraftBukkit and listed the results in Table 2. Without considering 29 developers whose region is still unknown, the remaining 67 comes from 11 countries. Among which the US occupies the majority of 32 developers. Obviously, most of them are European and American countries. Dinnerbone from UK and grum from Netherlands had made the largest contribution.

Table 2 region composition of CraftBukkit

region	Developers count
USA	32
UK	7
Germany	6
Canada	4
Netherlands	4
Sweden	3
Australia	2
Estonia	1
Pakistan	1

Austria	1
Belgium	1
unknown	29

There are 40 dominators in CraftBukkit. Comparing to region composition mentioned above, although the United States has advantages over developer counts, it's not the main domination country. In addition, after observing the same-region and cross-region dominant diagrams respectively (Figure6-7), it can be found that developers came from different countries had much more complex dominant relationship. Taking sk89q from the United States as an example, he had eight cross-region dominant relationships while the corresponding number of the same-region diagram is only one. We thought it can also be interpreted as exotic attraction.

Law 3: developers from different regions are more likely to have dominant relationship.

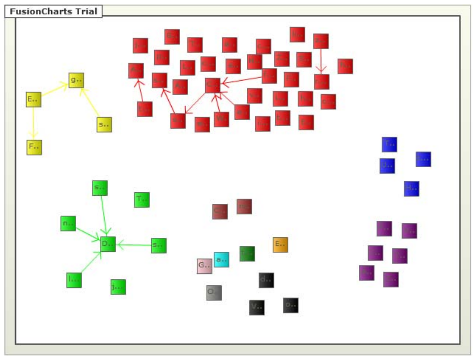

Figure 5. the same-region dominant diagram

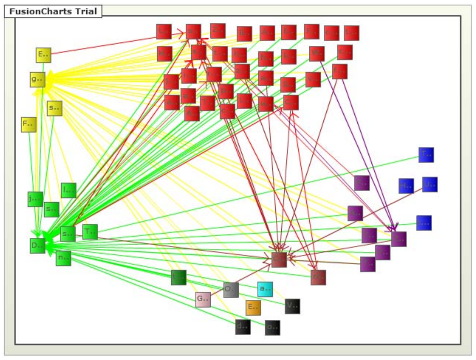

Figure 6. the cross-region dominant diagram

V. CONCLUSION

In this paper, we examined developers' contribution characteristics of CraftBukkit project based on visual analysis techniques. We examined the influencing factors of developers' contribution from the perspective of code and found that the code which newcomers started to contribute with more people engaged in would lead to less contribution in some degree; we applied barcode visualization in time characteristics of contribution and found that there's a relation between developers' early and later period contribution; we studied region characteristic of contribution using dominant graph and found that developers with different regions were more likely to have dominant relationship.

About the future research, we hope to carry out more case studies on varieties of open source software. Also, we want to know whether there is a similar or diametrically opposite conclusion in the area of commercial software.

REFERENCES

[1] M.M.Lehman. Programs, Life Cycles, and Laws of Software Evolution. Proceedings of the IEEE , Volume 68 Issue 9, 1060-1078, 1980

[2] Michael W.Godfrey, Qiang Tu. Evolution in open source software: A case study. Proceedings of the International Conference on Software Maintenance, 2000

[3] Jiangwei Wu, Richard C.Holt, Ahmed E.Hassan. Empirical Evidence for SOC Dynamics in Software Evolution. Software Maintenance, 2007

[4] Minghui Zhou, Audris Mockus. What Make Long Term Contributors: Willingness and Opportunity in OSS Community. ICSE'12, pp.518-528

[5] Narayan Ramasubbu, Marcelo Cataldo, Rajesh Krishna Balan, James Herbsleb. Configuring Global Software Teams: A Multi-Company Analysis of Project Productivity, Quality, and Profits. ICSE'11, pages 261-270

[6] Christian Bird, Nachiappan Nagappan. Who? Where? What? Examining Distributed Development in Two Large Open Source Projects. MSR 2012, Zurich, Switzerland

[7] Diomidis Spinellis. Global Software Development in the FreeBSD Project. Proceedings of the 2006 international workshop on Global Software Development for the practitioner, pages 73-79

[8] Michele Lanza University of Bern, Switzerland. The Evolution Matrix: Recovering Software Evolution using Software Visualization Techniques. IWPSE '01 Proceedings of the 4th International Workshop on Principles of Software Evolution, 2001.

[9] Jingwei Wu, Richard Holt, Ahmed Hassan. Exploring Software Evolution Using Spectrographs. 11th Working Conference on Reverse Engineering, 2004.

[10] Richard Wettel, Michele Lanza. Visual Exploration of Large-Scale System Evolution. WCRE, 2008.

[11] Alexandru Telea, David Auber. Code Flows: Visualizing Structural Evolution of Source Code. Eurographics/IEEE-VGTC Symposium on Visualization, 2008.

[12] Danny Holten, Jarke J. van Wijk. Visual Comparison of Hierarchically Organized Data. Computer Graphics Forum, Volume 27, 2008.

[13] Frank Steinbruchner, Claus Lewerentz. Representing development history in software cities. SOFTVIS, 2008.

[14] Abram Hindle, Zhen Ming Jiang, Walid Koleilat, Michael W.Godfrey, Richard C.Holt. YARN: Animating Software Evolution. Visualizing Software for Understanding and Analysis, 2007.

[15] Michael Ogawa, Kwan-Liu Ma. Software Evolution Storylines. SoftVis, 2010.

[16] Michael Ogawa, Kwan-Liu Ma. code_swarm : A Design Study in Organic Software Visualization. SOFTVIS, 2008.

[17] Michael Ogawa, Kwan-Liu Ma. StarGate: A Unified, Interactive Visualization of Software Projects. Visualization Symposium, 2008.

[18] Georgios Gousios, Eirini Kalliamvakou, Diomidis Spinellis. Measuring developer contribution from software repository data. Mining Software Repositories, 2008, pages 129-132.