Dynamics of Open-Source Software Developer’s Commit Behavior: An Empirical Investigation of Subversion

Yuta Ma  
State Key Laboratory of Software Engineering, Wuhan University  
Wuhan 430072, China  
ytma@whu.edu.cn

Yang Wu  
Information Center, LingYun Science and Technology Group Co., Ltd.  
Wuhan 430040, China  
awuyangc@163.com

Youwei Xu  
Information Center, Wuhan Housing Security and Management Bureau  
Wuhan 430015, China  
xuyouweiz@163.com

ABSTRACT

Commit is an important operation of revision control for open-source software (OSS). Recent research has been pursued to explore the statistical laws of such an operation, but few of those papers conduct empirical investigations on commit interval. In this paper, we investigated the dynamics of software developers’ collective commit behavior in terms of the distribution of commit intervals, and found that the data sets of project-level and file-level collective commit interval within both the lifecycle and each release of the projects under discussion roughly follow power-law distributions. The implications of what we found for OSS research were outlined, which could provide an insight into understanding OSS development processes better.

Categories and Subject Descriptors

D.2.9 [Management]: Life cycle, Programming teams, Software configuration management; K.6.3 [Software Management]: Software development, Software maintenance.

General Terms

Measurement, Experimentation, Human Factors.

Keywords

Open-source software, Subversion, Commit interval, Power law.

1. INTRODUCTION

It has been recognized that the open collaborative development among world-wide software developers is a significant factor to the success of open-source software (OSS) projects [1]. Revision control (or version control) is essential for the organization of multi-developer projects. In software engineering, the main role of revision control is to track and provide control over changes to source code [2]. As we know, OSS developers often use revision control tools such as CVS (Concurrent Versions System), Subversion (SVN) and Git to manage and maintain different types of files stored in a (code) repository that hosts an OSS project.

With these tools for revision control, software developers check out a given revision of a file or the most recent files from a repository to their local workspace of integrated development environments, and commit changes to the file(s) with related short comments/messages to the repository. So, the development of an OSS project is comprised of committing code contributions to a repository that hosts the project in this sense [3]. In order to understand OSS development and maintenance processes better, a number of papers have been pursued to investigate such an important operation. However, previous work [3] [4] [5] focused mainly on the distribution of commit size, to the best of our knowledge, few of them conducted empirical research on the dynamics of software developer’s commit behavior.

Human behavior, as one of the significant issues in science, has a history of about one century since the time of Watson [6]. In fact, the development of an OSS project could be deemed as a collaborative process of software developers’ collective commit behavior [7], but few researchers have yet investigated it from the perspective of mining OSS repositories [8]. So, the main purpose of this paper is to gain a deeper understanding of statistical laws for software developers’ collective commit behavior in centralized revision control systems. Based on an empirical investigation, we hope our research outcomes could be a new starting point for investigating how we can improve software quality and collective software development efficiency further.

The remainder of this paper is structured as follows. Section 2 introduces related work. Section 3 presents the analysis method we followed and preliminary results. Section 4 discusses the implications of what we found for OSS development and maintenance. Finally, Section 5 concludes this paper.

2. RELATED WORK

Commit size describes the probability that a given commit is of a particular size [5]. In 2008, the commit size in terms of the number of files committed for a revision was found to follow a Pareto distribution [4]. One year later, Arafat et al. found that the commit size in terms of source lines of code follows a power-law distribution [3]; similarly, the distribution of the commit size in terms of lines of code was confirmed to be best described by a generalized Pareto model [6]. The distribution of commit size with a long tail indicates that software developers might carry out large-size commits, though they are less likely to occur. Compared with the previous work, in this paper we focus on the general distribution of commit intervals.

Based on the increasing evidence from communication to entertainment and work patterns, Barabási et al. found that the timing of many human activities within these fields follow non-
Poisson statistics, characterized by bursts of rapidly occurring events separated by long periods of inactivity [9]. Interestingly, such heavy-tailed distributions of inter-event times have also been demonstrated in computer science. For example, the time interval between consecutive visits by a selected user to a given website is best approximated with a power-law distribution, in contrast to the exponential expected for Poisson processes [10]. However, there are few representative statistical analyses on the dynamics of software developer’s commit behavior according to appropriate data sources such as SourceForge and Google Code.

3. DATA ANALYSIS AND RESULTS

3.1 Metrics for Commit Behavior

For an OSS project under centralized revision control, a revision is a snapshot of its repository at a particular moment in time, and each successive commit increases the revision number by one [11].

**Definition 1.** Project-level collective commit interval (PLCCI) is the time difference (or waiting time) between two consecutive revisions in the repository of an OSS project.

For a file stored in the repository, a revision is basically a file that is modified when compared to the previous version. The history of file revision is all the information collected about a given file as it changes over time. Such information would be valuable for the research on file evolution, bug fixing and code refactoring.

**Definition 2.** File-level collective commit interval (FLCCI) is the time difference between two consecutive versions in the revision history of a file.

3.2 Data Collection

Table 1 shows a brief introduction to these four projects that were analyzed as of October 12, 2012, including the number of releases, the number of class files, the total number of commits analyzed, and the number of committers.

| Project | Description | Release | Class | Commit | Committer |
|---------|-------------|---------|-------|--------|-----------|
| POI     | APIs for file processing | 11      | 2,438 | 8,588  | 10        |
| Tomcat  | Servlet container | 10      | 1,980 | 14,481 | 16        |
| Struts2 | Framework for web apps | 20      | 1,521 | 9,999  | 27        |
| Derby   | RDBMS       | 14      | 2,974 | 21,529 | 35        |

Our analysis method is based on case studies, so we selected four OSS projects written in Java on the Apache.org: Apache POI, Tomcat, Struts2, and Derby. These four projects from different application domains were chosen to be experimental subjects based upon that each project in question has been active for at least 3 years and attracts over 10 fixed software developers or committers to participate in. Table 1 shows a brief introduction to these four projects.

3.3 Data Processing

Popular functions such as power function, exponential function, polynomial function, and logarithmic function were utilized to fit release-level and lifecycle-level data sets, so as to indentify the best fitting curve and its corresponding function expression. Because the projects in question are still evolving, we use a long period of development time (over three years) to approximate the lifecycle of each project. Although in statistics a distribution can be represented as a probability distribution function (PDF), we used a cumulative distribution function (CDF) to reduce noise levels during the estimation of the scaling exponent of power function with the method introduced in [12].

3.4 Preliminary Results

The scattered points of lifecycle-level PLCCI in hours and in days are presented in Figure 1, where X axis denotes the length of a commit interval and Y axis represents the number of commit intervals whose lengths are greater than or equal to a given number. It is obvious from the log-log plot that most of the waiting times between two consecutive commits last several hours (i.e., more than 80% of commit intervals are less than one day), but only a few of them experience a long duration of waiting that exceeds one week.

![Figure 1. Distributions of lifecycle-level PLCCI](image)

Interestingly, all release-level data about PLCCI in hours can be also best fitted by power functions, implying a general pattern of collective commit behavior recurred within different stages in the development process of an OSS project. Due to the space limitations, in this paper we only give an example of POI to illustrate how the data of lifecycle-level and release-level PLCCI in hours were fitted (see Table 2, where $R^2$ is the goodness of fit).

Table 2. Example of fitting functions for lifecycle-level and release-level PLCCI in Apache POI

| Sample  | Logarithmic          | $R^2$  | Polynomial       | $R^2$  | Exponential       | $R^2$  | Power           | $R^2$  |
|---------|----------------------|--------|------------------|--------|-------------------|--------|-----------------|--------|
| Lifecycle | $y=1.506x+398.570$  | 0.926  | $y=0.015x^2+7.351x+779.360$ | 0.580  | $y=414.43e^{0.018x}$ | 0.941  | $y=2E+06e^{-2.273x}$ | 0.964  |
| 3.5-65  | $y=-0.366x^2+37.311$ | 0.757  | $y=-0.005x^2-0.996x+49.701$ | 0.909  | $y=50.849e^{0.022x}$ | 0.984  | $y=783.07e^{-1.125x}$ | 0.996  |
| 3.5-66  | $y=-0.183x^2+35.674$ | 0.485  | $y=-0.002x^2-0.743x+60.342$ | 0.791  | $y=37.623e^{0.017x}$ | 0.882  | $y=4874.9e^{-1.794x}$ | 0.992  |
| 3.5-final | $y=-0.184x^2+29.550$ | 0.652  | $y=-0.002x^2-0.591x+42.693$ | 0.865  | $y=37.143e^{0.019x}$ | 0.969  | $y=755.55e^{-1.008x}$ | 0.994  |
| 3.6     | $y=-0.364x^2+36.344$ | 0.550  | $y=-0.008x^2-1.304x+55.159$ | 0.799  | $y=39.381e^{0.028x}$ | 0.914  | $y=423.64e^{-1.056x}$ | 0.973  |
| 3.7-61  | $y=-0.079x^2+28.038$ | 0.380  | $y=-0.001x^2-0.341x+48.564$ | 0.654  | $y=26.011e^{0.008x}$ | 0.878  | $y=2175.8e^{-1.242x}$ | 0.974  |
| 3.7-62  | $y=-0.103x^2+14.430$ | 0.614  | $y=-0.001x^2-0.240x+20.934$ | 0.833  | $y=16.771e^{0.016x}$ | 0.909  | $y=256.65e^{-1.197x}$ | 0.979  |
| 3.7-63  | $y=-0.068x^2+12.583$ | 0.402  | $y=-0.001x^2-0.275x+20.531$ | 0.649  | $y=11.471e^{0.019x}$ | 0.810  | $y=240.95e^{-1.787x}$ | 0.963  |
After a class file was created in the SVN repository of a project, various software developers would modify it together and then commit changes to it to the repository. For the projects under discussion, we found that both lifecycle-level FLCCI at different time scales and 55 release-level data sets of FLCCI in days were best fitted by the similar law. The finding indicates that the commit intervals of frequently-modified class files are relatively short on average (i.e., more than 80% of commit intervals of these classes are less than 3 days), while only a minority of classes’ revisions need to wait for a very long time of years (see Table 3, where $r$ in the last row means power exponent without a unit of measurement), perhaps because they are trivial, deleted, or found to have hidden bugs that are not always easy to detect.

### Table 3. Key statistical results of FLCCI in days for class files

| Sample | Mean | 1st Quartile | 2nd Quartile | 3rd Quartile | 90th Percentile | 95th Percentile |
|--------|------|--------------|--------------|--------------|----------------|----------------|
| Tomcat | 52.88 | 0.04 | 6.99 | 60.80 | 173.59 | 260.85 |
| Struts2 | 81.36 | 0.01 | 1.09 | 61.59 | 242.01 | 477.46 |
| Derby | 128.48 | 1.76 | 23.09 | 145.19 | 286.28 | 623.44 |
| POI | 62.47 | 0.01 | 3.19 | 79.96 | 198.36 | 318.12 |
| release-$r$ | 2.516 | 2.149 | 2.385 | 2.591 | 2.723 | 2.905 |

### 4. IMPLICATIONS FOR OSS RESEARCH

Although the projects analyzed differ in class size and the number of committers, each project varies slightly in terms of the distribution of PLCCI. Such an indicator reflects to some extent the level of activity of a project developed by various software developers. We argue that power-law distributions of lifecycle-level and release-level PLCCI may be derived from the mode of centralized revision control as well as the basic principles of incremental development.

On one hand, SVN uses a centralized model where all the revision control functions take place on a shared server [2], and it stores the latest version of each file in a central repository, with backward-looking differences between two adjacent revisions [11]. Updates to HEAD of the trunk in a SVN repository committed by different committers are always finished quickly so as to ensure the normal collaborative development among different software developers. On the other hand, OSS projects at large follow the principles of incremental development. We found an interesting phenomenon that the waiting time between the last commit within the previous release and the first commit after the delivery of a new release is longer than normal values; moreover, it recurs within all 55 releases of the projects in question. This could be used to explain the occurrence of small probability events (with long waiting time) in power-law distribution of PLCCI.

Division as well as cooperation is the basic principle of modern software development. Power-law distributions of lifecycle-level and release-level FLCCI may be a consequence of a queuing process driven by human decision making based on priority. When a committer has/receives multiple changes to be committed, he/she has to assess and prioritize the changes to different class files, and then allocates time for the chosen files with high perceived priority, which can be simply estimated in terms of functional and structural importance. So, committers often commit changes to those class files with complex function and structure in short order, while the waiting times for two consecutive commits of inactive or trivial class files are very long.

### 5. CONCLUSIONS

We took four OSS projects on the Apache.org for example to investigate the general statistical laws for the distributions of lifecycle-level and release-level commit interval. The preliminary findings are 1) both lifecycle-level and release-level PLCCI roughly follow power-law distributions, and 2) the distributions of both lifecycle-level and release-level FLCCI are found to share similar laws. The future work is to investigate work patterns of active committers and its impacts on OSS development processes.

### 6. ACKNOWLEDGMENTS

This work is supported by the National Basic Research Program of China under Grant No. 2014CB340401 and the National Natural Science Foundation of China under Grant No. 61272111.

### 7. REFERENCES

[1] Crowston, K., Wei, K., Howison, J. and Wiggins, A. Free/Libre Open Source Software Development: What We Know and What We Do Not Know. *ACM Computing Surveys*, 44, 2 (Feb. 2012), Article No. 7.

[2] O’Sullivan, B. Making sense of revision-control systems. *Communications of the ACM*, 52, 9 (Sep. 2009), 56-62.

[3] Arafat, O. and Riehle, D. The Commit Size Distribution of Open Source Software. In *Proc. |HICSS’09* (Hawaii, USA, January 5-8, 2009). IEEE Computer Society Press, New York, NY, 2009, 1-8.

[4] Hattori, L. and Lanza, M. On the nature of commits. In *Proc. |EVL’08* (L’Aquila, Italy, September 15-16, 2008). IEEE Computer Society Press, New York, NY, 2008, 63-71.

[5] Kolassa, C., Riehle, D. and Salim, M. A Model of the Commit Size Distribution of Open Source. In *Proc. |SOFSEM’13* (Špindleruv Mlyn, Czech Republic, January 26-31, 2013). Springer-Verlag, Heidelberg, Baden-Württemberg, 2013, 52-66.

[6] Watson J. Psychology as the Behaviorist Views It. *Psychological Review*, 20, 2 (Feb. 1913), 158-177.

[7] Lin, S., Ma, Y. and Chen, J. Empirical Evidence on Developer’s Commit Activity for Open-Source Software Projects. In *Proc. |SEKE’13* (Boston, USA, June 27-29, 2013). Knowledge Systems Institute, Skokie, Illinois, 2013, 455-460.

[8] Baysal, O., Holmes, R. and Godfrey M. Mining usage data and development artifacts. In *Proc. |MSR’12* (Zurich, Switzerland, June 2-3, 2012). IEEE Computer Society Press, New York, NY, 2012, 98-107.

[9] Barabási, A.-L. The origin of bursts and heavy tails in human dynamics. *Nature*, 435, 7039 (May. 2005), 207-211.

[10] Dezső, Z., Almaas, E., Lukács, A., Rácz, B., Szakadát, I. and Barabási, A.-L. Dynamics of information access on the web. *Physical Review E*, 73, 6 (Jun. 2006), 066132.

[11] Collins-Sussman B. The subversion project: building a better CVS. *Linux Journal*, 2002, 94 (Feb. 2002), Article No. 3.

[12] Clauset, A., Shalizi, C. and Newman, M. Power-law distributions in empirical data. *SIAM Review*, 51, 4 (Apr. 2009), 661-703.