Understanding Source Code Comments at Large-Scale

Hao He
Peking University, China
heh@pku.edu.cn

ABSTRACT

Source code comments are important for any software, but the basic patterns of writing comments across domains and programming languages remain unclear. In this paper, we take a first step toward understanding differences in commenting practices by analyzing the comment density of 150 projects in 5 different programming languages. We have found that there are noticeable differences in comment density, which may be related to the programming language used in the project and the purpose of the project.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**.

KEYWORDS

Source Code Comments, Comment Density, Empirical Study

ACM Reference Format:
Hao He. 2019. Understanding Source Code Comments at Large-Scale. In *Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19), August 26–30, 2019, Tallinn, Estonia.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3338906.3342494

1 PROBLEM AND MOTIVATION

Source code comments constitute an important part of any software, which help people understand code and facilitate software maintenance [14, 15]. To understand how programmers write comments and find insights for improving software practices, existing studies have analyzed comments from various perspectives, such as ratio of comments [7], comment code co-evolution [2], and the purpose of comments [10]. However, these studies are often limited in one programming language, one or several projects and one specific aspect of code comments. Meanwhile, the very basic pattern of writing comments across domain and language remains unclear, while it may greatly help software projects understand their position and adjust their practices accordingly. The main reason might be that it is not easy to access sufficient projects to make a comparison. In particular, we may not be able to access a large amount of projects and the effort to collect the needed data is significant. Recently, the rise of large open source platforms such as GitHub and the emergence of open source project databases like GHTorrent [3] and World of Code [6] enable large scale analysis of software projects. Therefore, we set up to conduct a large scale investigation of code comments and expect to help practices in various ways, e.g., defining benchmark for comment density, locating where to comment, and generating comments for code.

ESEC/FSE '19, August 26–30, 2019, Tallinn, Estonia

ACM ISBN 978-1-4503-5572-8/19/08.
https://doi.org/10.1145/3338906.3342494

2 BACKGROUND AND RELATED WORK

Programmers frequently write comments along with source code. As a result, code comments form an important part of documentation, providing additional information not immediately visible from source code. Studies have shown that reading source code with comments aid with program comprehension [14, 15]. Further research reveals that the quality of comment itself, especially the consistency between source code and comments, is crucial for avoiding software bugs and improving maintainability [2, 12].

Because of the important role of comments in program comprehension, software quality and software maintenance, there have been a number of studies that analyze comments in existing software projects [1, 2, 4, 5, 9, 10]. However, existing studies either focus on one programming language [4, 10] or one specific application domain(e.g. operating systems [9]), or consider only one specific dimension of comments (e.g. comment density [1], links in comments [5]). To the best of our knowledge, no existing research has focused on analyzing commenting practices and their differences in a large number of heterogeneous projects.

3 APPROACH

We take an initial step towards understanding commenting practices across projects by addressing the following research questions:

- RQ1: Do projects practice commenting differently?
- RQ2: What may cause the differences?

3.1 Selection of Open Source Projects

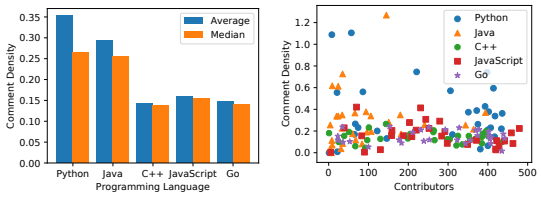
We choose five most popular programming languages among 1000 most starred Repositories on GitHub (JavaScript, Java, C++, Python and Go at the time of April 2019) and collect 30 most starred repositories for each programming language. This is a relatively small dataset for preliminary analysis and we plan to use the World of Code [6] database in the future.

3.2 Analysis of Comment Density

To answer RQ1, we begin from one simple metric: comment density, which has also been used to measure software maintainability [7] and quality [11]. We plan to investigate more sophisticated metrics in the future, such as vocabulary used in comments and distribution of comments over program structures.

We define comment density of a project as follows:

$$\text{Comment Density} = \frac{\text{Line of Comments}}{\text{Line of Code}} \quad (1)$$



(a) Comment density in different programming languages

(b) Relationship between comment density and # of contributors

Figure 1: Figures for Comment Density Analysis

For each project, we count its lines of code and lines of comment of its major programming language.

To answer RQ2, we propose the following hypothesis based on existing literature and practical experiences:

(1) H1: The programming language used in a project may affect its comment density.
(2) H2: The purpose (domain) of a project may affect its comment density.
(3) H3: Team size may affect the comment density of a project because more people need to read the code.

4 RESULTS

RQ1: Do projects practice commenting differently? We find that comment density varies greatly in the 150 collected projects ($avg = 0.2124, stddev = 0.1807, max = 1.2691, min = 0.003$, excluding projects with no code at all). The most heavily commented project has more lines of comments than source code (which is java-design-patterns[1], with 29414 lines of code and 37329 lines of comments). On the other hand, comments in some projects are extremely scarce, e.g., Font-Awesome[2], with 73808 lines of code and 240 lines of comments. The results suggest that projects do have different commenting practices.

RQ2: What may cause the differences?

To confirm H1, we plot the average and median comment density for different programming languages (Figure 1a). Since the distribution is not normal, we conduct Wilcoxon signed-rank test on languages pairs and find that the comment density of Python and Java projects is significantly higher than C++, JavaScript and Go projects(See Table 1 for original p-values). One possible explanation is that there are widely adopted documentation generation tools for Java [8] and Python [13], which specify a given set of rules for programmers to write comments. The other three languages, however, have no widely adopted rules for writing comments.

To confirm H2, we manually inspect 30 Java projects and 30 JavaScript projects in the collected dataset. We identify three major purposes for which the project is used:

(1) **Software Reuse**. The project is a framework or a library, which provides functions or solutions for other people to reuse in their own applications (e.g. Vue.js[3], a progressive web framework for developers to build web applications).
(2) **Application**. The project is a complete and ready-to-use application for interested users (e.g. proxyee-down[4], an HTTP downloader implemented in Java).
(3) **Education**. The project is set up for educational purposes. Users of this project are supposed to understand and learn from the source code (e.g. Android-CleanArchitecture[5], an example to learn how to architect an Android application).

Table 1: Original p-Values of the Wilcoxon Signed-rank Test

	Python	Java	C++	JavaScript	Go
Python	1.0000	0.4420	0.0003	0.0034	0.0008
Java		1.0000	0.0027	0.0115	0.0043
C++			1.0000	0.7227	0.7562
JavaScript				1.0000	0.9764
Go					1.0000

Table 2: Average Comment Density by Project Purpose

	Education	Software Reuse	Application
Java	0.5751	0.2739	0.0641
JavaScript	0.2650	0.1760	0.1050

Table 2 summarizes average comment density of the three different type of projects, for Java and JavaScript respectively. Projects with educational purposes have the highest comment density, while projects which are ready-to-use applications have the lowest, and projects with software reuse purposes stay in the middle. One possible explanation is that, for educational projects, it is important to have enough comments so that most users can understand the code. For applications, only core developers need to read and understand its source code and only a minimum amount of comments are necessary. For reusable open source libraries and frameworks, users occasionally need to read its source code to understand its usage or find bugs, and thus they need to have a reasonable amount of comments. However, we have to point out that the dataset is too small to conduct any statistic significance tests. We plan to replicate on a larger dataset in the future.

To confirm H3, we plot the number of contributors along with comment density (Figure 1b). However, we fail to observe any correlation that supports H3. Further investigation is needed to reveal the relationship between comment density and team size.

5 CONCLUSION

We take a first step toward understanding the differences of source code comments across various projects. We have found that there are indeed noticeable differences in comment density of different projects, which may be related to the programming language used in the project and the purpose of the project. The result is promising and we plan to further investigate this problem in the future.

[1] https://github.com/iluwatar/java-design-patterns
[2] https://github.com/FortAwesome/Font-Awesome/
[3] https://github.com/vuejs/vue
[4] https://github.com/proxyee-down-org/proxyee-down
[5] https://github.com/android10/Android-CleanArchitecture

REFERENCES

[1] Oliver Arafat and Dirk Riehle. 2009. The comment density of open source software code. In *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Companion Volume*. 195–198. https://doi.org/10.1109/ICSE-COMPANION.2009.5070980

[2] Beat Fluri, Michael Würsch, Emanuel Giger, and Harald C. Gall. 2009. Analyzing the Co-evolution of Comments and Source Code. *Software Quality Journal* 17, 4 (Dec. 2009), 367–394. https://doi.org/10.1007/s11219-009-9075-x

[3] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: Github's data from a firehose. In *9th IEEE Working Conference of Mining Software Repositories, MSR 2012, June 2-3, 2012, Zurich, Switzerland*. 12–21. https://doi.org/10.1109/MSR.2012.6224294

[4] Dorsaf Haouari, Houari A. Sahraoui, and Philippe Langlais. 2011. How Good is Your Comment? A Study of Comments in Java Programs. In *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22-23, 2011*. 137–146. https://doi.org/10.1109/ESEM.2011.22

[5] Hideaki Hata, Christoph Treude, Raula Gaikovina Kula, and Takashi Ishio. 2019. 9.6 Million Links in Source Code Comments: Purpose, Evolution, and Decay. *CoRR* abs/1901.07440 (2019). arXiv:1901.07440 http://arxiv.org/abs/1901.07440

[6] Yuxing Ma, Christopher Bogart, Sadika Amreen, Russell Zaretzki, and Audris Mockus. 2019. World of Code: An Infrastructure for Mining the Universe of Open Source VCS Data. In *16th International Conference on Mining Software Repositories, MSR 2019*.

[7] P. Oman and J. Hagemeister. 1992. Metrics for assessing a software system's maintainability. In *Proceedings Conference on Software Maintenance 1992*. 337–344. https://doi.org/10.1109/ICSM.1992.242525

[8] Oracle. 2019. Javadoc. https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html. Accessed: 2019-06-05.

[9] Yoann Padioleau, Lin Tan, and Yuanyuan Zhou. 2009. Listening to programmers - Taxonomies and characteristics of comments in operating system code. In *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings*. 331–341. https://doi.org/10.1109/ICSE.2009.5070533

[10] Luca Pascarella and Alberto Bacchelli. 2017. Classifying code comments in Java open-source software systems. In *Proceedings of the 14th International Conference on Mining Software Repositories, MSR 2017, Buenos Aires, Argentina, May 20-28, 2017*. 227–237. https://doi.org/10.1109/MSR.2017.63

[11] Ioannis Stamelos, Lefteris Angelis, Apostolos Oikonomou, and Georgios L. Bleris. 2002. Code Quality Analysis in Open Source Software Development. *Information System Journal* 12, 1 (2002), 43–60. https://doi.org/10.1046/j.1365-2575.2002.00117.x

[12] Lin Tan, Ding Yuan, Gopal Krishna, and Yuanyuan Zhou. 2007. /*Icomment: Bugs or Bad Comments?*/. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles (SOSP '07)*. ACM, New York, NY, USA, 145–158. https://doi.org/10.1145/1294261.1294276

[13] The Sphinx team. 2019. Sphinx. http://www.sphinx-doc.org/en/master/. Accessed: 2019-06-05.

[14] T. Tenny. 1988. Program Readability: Procedures Versus Comments. *IEEE Trans. Softw. Eng.* 14, 9 (Sept. 1988), 1271–1279. https://doi.org/10.1109/32.6171

[15] S. N. Woodfield, H. E. Dunsmore, and V. Y. Shen. 1981. The Effect of Modularization and Comments on Program Comprehension. In *Proceedings of the 5th International Conference on Software Engineering (ICSE '81)*. IEEE Press, Piscataway, NJ, USA, 215–223. http://dl.acm.org/citation.cfm?id=800078.802534