Classifying Code Comments in Java Mobile Applications

Luca Pascarella
Delft University of Technology, The Netherlands
L.Pascarella@tudelft.nl

ABSTRACT
Developers adopt code comments for different reasons such as document source codes or change program flows. Due to a variety of use scenarios, code comments may impact on readability and maintainability. In this study, we investigate how developers of 5 open-source mobile applications use code comments to document their projects. Additionally, we evaluate the performance of two machine learning models to automatically classify code comments. Initial results show marginal differences between desktop and mobile applications.

CCS CONCEPTS
• Software and its engineering → Maintaining software;

KEYWORDS
Android, Mining Software Repositories, Code Comments

ACM Reference Format:

1 INTRODUCTION
During software development, software engineers make several choices for forging computer programs [5] and to document their rationals, developers write code comments [10]. Past researches demonstrate that code comments are crucial to enhance program readability and maintainability [3]. Despite unaligned documentation, it may exacerbate maintenance [4, 9], researchers globally agree that having a generous commented code is a good practice [2, 6].

Considering all comments the same may bring to incorrect evaluations especially when code comments are used to perceive the quality of inspected codes, for example, by measuring the code/comment ratio [2, 6]. In a recent study, Pascarella et al. confirmed this limitation arguing that code comments contribute to different meanings [8]. They proposed a novel taxonomy to classify Java comments, investigated how developers of OSS systems use comments, and experimented with automatically classifying code comments.

In this study, we aim at corroborating and possibly improving the current knowledge about the use of code comments in mobile apps. Particularly, we (1) measured the distribution of code comments in the given taxonomy and (2) evaluated the performance of two machine learning models to automatically classify code comments. For this purpose, we inspected 325 Java files of 5 open-source Android mobile apps and we manually classified up to 2,100 comment blocks comprising more than 4,500 lines.

Our results confirm the suitability of the proposed taxonomy in the context of mobile apps. We discovered only marginal differences between desktop and mobile apps. Finally, even though the performance of the machine learning classifiers decreases in the context of a cross-project training we detected a promising capability in reusing the provided training set.

2 METHODOLOGY
The intention of this work denotes and extends the goal of the study conducted by Pascarella et al. aimed at understanding the purpose of the code comment written by developers [8]. Particularly, this study focuses on code comments of open-source Android mobile apps by verifying the generalizability of the proposed approach.

Research questions. To this aim, we observed that Pascarella et al. mainly focus on Java desktop apps [8]. Although the desktop and mobile apps share the same programming languages, developers could adopt the same development approach. A study showed the opposite [11]. Consequently, to understand how Android developers use code comments we defined our first research question:

Q1. How often does each category occur in OSS Android apps using the Pascarella et al. taxonomy?

Then, we investigate to what extent the automated models proposed by Pascarella et al. can be generalized in a cross-project approach. This leads to our second research question:

Q2. How effective are the proposed machine learning models in classifying code comments in OSS Android apps?

Project selection. To conduct our analysis we selected 5 heterogeneous Android apps written in Java programming language. They are all open-source projects available through Google Play, hosted by GitHub, and with different size and scope. Table 2 summarizes the characteristics of the selected systems reporting for each project the GitHub link, the number of commits, the number of contributors, and the number of Java lines.

<table>
<thead>
<tr>
<th>Project</th>
<th>GitHub</th>
<th>Google Play</th>
<th>Commits</th>
<th>Cont.</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFWall+</td>
<td><a href="https://goo.gl/ux7hAj">https://goo.gl/ux7hAj</a></td>
<td><a href="https://goo.gl/382h">https://goo.gl/382h</a></td>
<td>2,346</td>
<td>21</td>
<td>24k</td>
</tr>
<tr>
<td>Amazon File Manager</td>
<td><a href="https://goo.gl/367AJ">https://goo.gl/367AJ</a></td>
<td><a href="https://goo.gl/V5jv8l">https://goo.gl/V5jv8l</a></td>
<td>2,913</td>
<td>89</td>
<td>41k</td>
</tr>
<tr>
<td>AntennaPod</td>
<td><a href="https://goo.gl/5XhI4F">https://goo.gl/5XhI4F</a></td>
<td><a href="https://goo.gl/Yrn5CB">https://goo.gl/Yrn5CB</a></td>
<td>2,913</td>
<td>103</td>
<td>63k</td>
</tr>
<tr>
<td>ownCloud</td>
<td><a href="https://goo.gl/566vHC">https://goo.gl/566vHC</a></td>
<td><a href="https://goo/gl/NwVidp">https://goo/gl/NwVidp</a></td>
<td>6,897</td>
<td>76</td>
<td>69k</td>
</tr>
</tbody>
</table>

Sample validity. To establish a statistical significant sample of Java files used to measure to what extent Java code comments are
We report the top whose purpose was not covered by the provided taxonomy.

Figure 1: Comparison of code comments frequency.

Table 2: Performance of the Random Forest classifier

<table>
<thead>
<tr>
<th>Categories</th>
<th>P/R</th>
<th>10-fold</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>P</td>
<td>0.98</td>
<td>0.63</td>
<td>0.70</td>
<td>0.94</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>Notice</td>
<td>P</td>
<td>0.94</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Under</td>
<td>P</td>
<td>0.99</td>
<td>1.00</td>
<td>0.75</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>development</td>
<td>R</td>
<td>0.99</td>
<td>0.33</td>
<td>0.30</td>
<td>0.80</td>
<td>0.80</td>
<td>0.80</td>
</tr>
<tr>
<td>Style</td>
<td>P</td>
<td>0.84</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>&amp; IDE</td>
<td>R</td>
<td>0.91</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Metadata</td>
<td>P</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Discarded</td>
<td>P</td>
<td>1.00</td>
<td>0.95</td>
<td>0.73</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

4 THREATS TO VALIDITY AND CONCLUSION

Awe of limitations of our dataset (low number of apps and files) and the taxonomy validity (new categories may emerge) we plan to extend this study by considering a higher number of projects besides a revisited taxonomy validation. Additionally, we plan to overcome the limitation of the cross-project train/test model by creating a representative dataset of mobile apps comments. This dataset may be used to pre-filter code comments and improve the performance of self-admitted technical debt methods.

With our preliminary study, we observed that in both desktop and mobile projects code comments contain valuable information for supporting software development. However, mobile apps developers tend to use code comments for a different purpose. For example, the high percentage of Commented Code category may represent a bad practice, such as with negative consequence on readability and maintainability. Finally, we observed a limitation of supervised classifiers when applied to cross-project validation.

REFERENCES