Classifying code comments in Java Mobile Applications

Pascarella, Luca

DOI
https://doi.org/10.1145/3197231.3198444

Publication date
2018

Document Version
Accepted author manuscript

Published in
Conference on Mobile Software Engineering and Systems

Citation (APA)

Important note
To cite this publication, please use the final published version (if applicable). Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.
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 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 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 experiment 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:

RQ1. 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:

RQ2. 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 Google PLAY link, the number of commits, the number of contributors, and the number of Java files.

Table 1: Overview of the projects used in this study

<table>
<thead>
<tr>
<th>Project</th>
<th>GitHub</th>
<th>Google Play</th>
<th>Commits</th>
<th>Cont.</th>
<th>Lines</th>
</tr>
</thead>
</table>

Sample validity. To establish a statistical significant sample of Java files used to measure to what extent Java code comments are...
used by mobile apps developers we considered all 2108 Java files, we
deﬁned a conﬁdence level of 95%, and we set a conﬁdence interval
of 5%. We obtained a representative sample of 325 Java ﬁles. This
sample creates a reasonable large dataset composed of about 2,100
block of comments and more than 4,500 lines of comments.

Manual classiﬁcation. To answer our RQ1 we manually inspec-
ted and classiﬁed all comments of the sampled Java ﬁles. To
this aim, we used the deﬁnitions of the taxonomy proposed by
Pascarella et al. [8]. Such a taxonomy is composed of 22 categories
organized into 2 levels. The ﬁrst level discurs 6 top categories while
the second level details 16 deﬁnitions (inner categories). Thereafter,
we used the COMMean tool [8] to support the classiﬁcation process and
reduce human errors. Moreover, we annotated possible comments whose
purpose was not covered by the provided taxonomy.

Classiﬁcation technique. To answer our RQ2 we repeated the
settings of Pascarella et al. by applying only a subset of machine
learning classiﬁers. Particularly, we used 2 well-known classes of
supervised machine learning algorithms based on probabilistic
classiﬁers and decision tree algorithms, Naïve Bayes Multinominal
and Random Forest, respectively. These algorithms rely on diverse
assumptions aimed at reacting to different execution speeds and
overﬁtting ability. Finally, to improve the performance we applied
both data balancing and multi-collinearity corrections.

Classiﬁcation evaluation. To evaluate the performance of the
achieved results we relied on precision and recall, which are based
on the evaluation of different combinations of True Positive, False
Positive, and False Negative. Finally, we evaluated the performance
considering 2 diﬀerent evaluation techniques: (1) a preliminary
evaluation adopting the 10-fold cross-validation, then (2) intersect-
datasets of diﬀerent projects to train and test the model and
overcome the limitation imposed by data availability [1].

3 RESULTS
We report the top categories’ results (a detailed report is available
online with a new dataset of classiﬁed mobile apps comments [7]).
RQ1 results. Figure 1 compares the distributions of comments for
desktop and mobile apps. An appreciable diﬀerence is evident
for Under Dev. category that for mobile apps is more than the dou-
ble. It suggests that these developers frequently deal with unstable
features. While this represents a dynamic development it is also a
sign of self-admitted technical debts. At same time Purpose cate-
gory suggests that mobile apps developers resort to code comments
to document their code more than desktop apps developers.

RQ2 results. Table 2 shows only the performance of the better
classiﬁer tested considering precision (P) and recall (R) for 10-fold
and cross-project validation. While 10-fold conﬁrms considerable
performance, cross-project validation suﬀers. This drop is due to
a well-known limitation of a supervised algorithm that does not
tolerate project-speciﬁc key terms.

Despite limitations, this classiﬁcation method may still help the
development life cycle providing an overview of the code quality.

4 THREATS TO VALIDITY AND CONCLUSION
Aware 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 de-
tevelopers tend to use code comments for a diﬀerent purpose. For
e.g., the high percentage of Commented Code category may rep-
resent 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
2012. Content Classiﬁcation of Development Emails. In ICSE.
as a key factor in maintenance productivity: a case study. In ICSE.
Comments in PostgreSQL. In Workshop on Mining Software Repositories.
"Whys": Discovering design rationale using text mining: An algorithm perspective.
maintainability. In Software Maintenance.
10838253.
open-source software systems. In Mining Software Repositories (MSR).