Information and Software Technology 88 (2017) 148–158

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

On code reuse from StackOverflow: An exploratory study on Android apps

Rabe Abdalkareem[a,*], Emad Shihab[a], Juergen Rilling[b]

[a] Data-driven Analysis of Software (DAS) Lab, Concordia University, Montreal, Canada
[b] Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada

ARTICLE INFO

Article history:
Received 12 May 2016
Revised 16 March 2017
Accepted 16 April 2017
Available online 17 April 2017

Keywords:
StackOverflow
Mobile app
Code reuse

ABSTRACT

Context: Source code reuse has been widely accepted as a fundamental activity in software development. Recent studies showed that StackOverflow has emerged as one of the most popular resources for code reuse. Therefore, a plethora of work proposed ways to optimally ask questions, search for answers and find relevant code on StackOverflow. However, little work studies the impact of code reuse from StackOverflow.

Objective: To better understand the impact of code reuse from StackOverflow, we perform an exploratory study focusing on code reuse from StackOverflow in the context of mobile apps. Specifically, we investigate how much, why, when, and who reuses code. Moreover, to understand the potential implications of code reuse, we examine the percentage of bugs in files that reuse StackOverflow code.

Method: We perform our study on 22 open source Android apps. For each project, we mine their source code and use clone detection techniques to identify code that is reused from StackOverflow. We then apply different quantitative and qualitative methods to answer our research questions.

Results: Our findings indicate that 1) the amount of reused StackOverflow code varies for different mobile apps, 2) feature additions and enhancements in apps are the main reasons for code reuse from StackOverflow, 3) mid-age and older apps reuse StackOverflow code mostly later on in their project lifetime and 4) that in smaller teams/apps, more experienced developers reuse code, whereas in larger teams/apps, the less experienced developers reuse code the most. Additionally, we found that the percentage of bugs is higher in files after reusing code from StackOverflow.

Conclusion: Our results provide insights on the potential impact of code reuse from StackOverflow on mobile apps. Furthermore, these results can benefit the research community in developing new techniques and tools to facilitate and improve code reuse from StackOverflow.

1. Introduction

A key premise of software development is to deliver high-quality software in a timely and cost-efficient manner. Code reuse has been widely accepted to be an essential approach to achieve this premise [1]. The reused code can come from many different sources and in different forms, e.g., third-party libraries [1], source code of open source software [2], and Question and Answer (Q&A) websites such as StackOverflow [3,4].

In recent years, the development of mobile apps has emerged to be one of the fastest growing areas of software development [5]. Some of the most common characteristics of mobile apps are: (1) they are developed by small teams [6], (2) they are developed by less experienced programmers [7] and (3) they are constrained by limited resources [7]. As a result of these constraints, mobile developers tend to resort frequently to Q&A websites such as StackOverflow for solutions to their coding problems [3,8]. A primary challenge with this type of code reuse is that programmers often resort to these code snippets in an ad-hoc manner, by copying-and-pasting these fragments in their own source code. However, the impact of this ad-hoc reuse on software processes, product quality, and mobile app development at large remains an open question.

There has been a plethora of work focusing on StackOverflow (e.g., [9,10]), code reuse (e.g., [1,11]) and mobile development (e.g., [12,13]), which studied all of these topics in isolation. However, vital questions about how mobile developers reuse code from StackOverflow remain unanswered. This is particularly important since 1) prior work showed that StackOverflow is very popular among mobile developers [3,8] and 2) as we report later in the paper, once code is reused from StackOverflow, it can potentially have a negative impact on the target mobile app.

* Corresponding author.
E-mail addresses: rab_abdu@encs.concordia.ca (R. Abdalkareem), eshihab@encs.concordia.ca (E. Shihab), juergen.rilling@encs.concordia.ca (J. Rilling).

http://dx.doi.org/10.1016/j.infsof.2017.04.005
0950-5849/

Therefore, in an effort to better understand code reuse from StackOverflow amongst mobile developers, we perform an exploratory study using quantitative and qualitative methods on 22 open source mobile apps. In particular, we answer three fundamental questions about code reuse from StackOverflow which are: (RQ1) Why is StackOverflow code reused in mobile apps? It is important to study why StackOverflow code is reused since it helps us understand the key reasons and tasks where mobile app developers resort to StackOverflow. (RQ2) At what point of time during the development process of mobile apps does code reuse from StackOverflow occur? Knowing when in the project's lifetime code is mostly reused from StackOverflow helps us better understand in what stage of the development mobile developers need the most help and resources. RQ3) Who reuses code from StackOverflow? Knowing who reuses code from StackOverflow can provide us with insights on the type of reuse (e.g., is it reused due to lack of experience or is it well thought-out reuse by experienced developers).

We find that the amount of reused StackOverflow code varies among mobile apps. Also, the main reasons for code reuse from StackOverflow are to implement new features, enhance existing functionality, refactor code, use APIs, and test source code (RQ1). Moreover, we observe that mid-age and older apps reuse StackOverflow code later on in their lifetime (RQ2). With regards to the experience of the developers who reuse code depends on the team/app size; more experienced developers reuse code in smaller teams/apps, while less experienced developers reuse code in larger teams/apps (RQ3). Finally, to shed light on the potential impact of reused StackOverflow code on mobile apps, we examine bug fixing commits of files that contain code reused from StackOverflow. We find that these files have a higher percentage of bug fixing commits after the introduction of reused StackOverflow code, indicating that reusing such code may negatively impact the quality of mobile apps.

The rest of this paper is organized as follows: Section 2 introduces a motivated example of our research. Section 3 sets up our case study. Section 4 discusses our preliminary analysis on how much reuse occurs in mobile apps. In Section 5, we report our case-study results. We discuss the implications of code reuse on mobile app quality in Section 6. Related work is presented in Section 7. Section 8 presents the threats to validity and Section 9 concludes our study.

2. Motivating example

Existing research (e.g., [3,8]) has shown that developers of mobile apps resort to StackOverflow for help (e.g., to resolve implementation problems or examine best coding practices). In contrast to this existing work, the objective of our research is to study code reuse from StackOverflow during the implementation or maintenance of mobile apps.

In what follows, we show the motivation for our research, by describing a real world example, where code has been reused from StackOverflow in the WordPress[1] Android app (we also elaborate more on how much reuse is done later on in the paper). Fig. 1a shows an answer post with a code snippet which has been posted on StackOverflow[2] to provide a solution for the use of the getCheckedItemCount() method in the ListView class of the Android API, with the getCheckedItemCount() method returning the number of items currently selected in the list.

The post describes a compatibility issue that arises when the method is used in older Android API versions (prior to version 11). In the WordPress example, a developer encountered the same problem using the ListView class and resorted to the code solution posted on StackOverflow. Fig. 1b shows details of the actual commit that reuses the code from StackOverflow, which in this case even includes the link to the original StackOverflow post. In addition, using the timestamps of the commit and the StackOverflow post, we are able to determine that the code was committed in the WordPress for Android project after it was posted on StackOverflow.

Being able to identify that StackOverflow code exists in a mobile app is important for several reasons: first, the reused code can be considered third-party code that may negatively impact the project. In particular, given that the origin of these code snippets is unknown, the reused code needs to be carefully reviewed since it may be incomplete and/or may have been developed for a different context. Second, although we do not address it in this paper, using StackOverflow code can lead to potential license violations in the mobile app reusing the code snippet.

3. Case study setup

The goal of our paper is to perform an exploratory study on the reuse of StackOverflow code in mobile apps. To perform our study, we extract code snippets from StackOverflow in order to examine the code reuse. We then select 22 open source mobile apps and examine their source code for potential code reuse from StackOverflow. Fig. 2 provides an overview of our approach. Once we identify the reused code, we analyze our dataset and answer our research questions.

3.1. Building StackOverflow code snippets corpus

To perform our study, we started by downloading and extracting code snippets from StackOverflow. We obtained the StackOverflow data dump (published March 16, 2015) in XML format. The data dump contained 24,120,523 posts, including 8,978,719 questions and 15,141,804 answers. Each discussion includes a question and zero or more answer posts and their meta data *(e.g., body, creation date and number of votes)*. Questions are typically tagged with terms describing the categories under which these Q&A discussions are grouped.

During the next processing step, we extract the relevant discussions for a study context. Since we focus on Android open source apps, we extracted all discussions related to the "Java" tag (810,071 discussions). We then further filtered these discussions to ensure that they also contain the "Android" tag, which left us with 106,861 discussions related to Java and Android. Since we are only interested in code reuse from StackOverflow, we further limited our dataset to only consider discussions that contain source code. For our analysis, we focused on the source code in the answers of the StackOverflow discussions, since code in questions is not likely to be reused. This further reduced the set of discussion posts to 53,683. Since prior work suggested the use of highly voted code snippets, we only focused on code snippets in answers that have a high number of votes [10]. We therefore examined the average votes for posts with their code snippets size (P_{size}) within our 53,683 selected StackOverflow posts where, $P_{size} \leq 5$ *lines*, $P_{size} > 5 \cap P_{size} < 30$ *lines* and $P_{size} \geq 30$ *lines*. We found that snippets with 30 or more lines had the highest average votes with 2.02 while snippets with ≤ 5 *lines*, $> 5 \cap < 30$ *lines* have lower average numbers of votes (1.85 and 1.69, respectively). Therefore we decided to focus only on the posts which contained snippets with at least 30 lines of code because 1) they have the highest average votes, i.e., StackOverflow users find them the most helpful, 2) smaller code snippets would result in too many false positives (e.g., it may flag common and simply code such as `if` and `for` loop statements), 3) choosing 30 lines of code will decrease the size of the corpus and

[1] https://github.com/wordpress-mobile/WordPress-Android.

[2] http://stackoverflow.com/questions/12330660/whats-the-equivilent-of-getcheckeditemcount-for-api-level-11.

The accepted answer didn't work for me (always returns 0), I had to use the following code:

```
public static int getCheckedItemCount(ListView listView)
   {
      if (Build.VERSION.SDK_INT >= 11) return listView.getCheckedItemCount();
      else
      {
         int count = 0;
         for (int i = listView.getCount() - 1; i >= 0; i--)
            if (listView.isItemChecked(i))   count++;
         return count;
      }
}
```

edited Jun 28 '13 at 14:56 answered Jun 28 '13 at 14:44

Takhion

(a) Code snippet from Stack Overflow

```
SHA1 ID: a6e3669c0101455e827e65c5c191fe7a0217d8bb
Author: Dan Roundhill <dan@automattic.com>  2013-08-19 15:13:53
Message: ListView.getCheckedItemCount() is a v11 or higher api. Added solution from
http://stackoverflow.com/questions/12330660/whats-the-equivilent-of-getcheckeditemcount-for-
api-level-11
 that loops through the list to get the checked item count on older devices.

@@ -392,4 +393,16 @@ public class SelectCategoriesActivity extends SherlockListActivity {
+
+   private int getCheckedItemCount(ListView listView)
+   {
+      if (Build.VERSION.SDK_INT >= 11) return listView.getCheckedItemCount();
+      else
+      {
+         int count = 0;
+         for (int i = listView.getCount() - 1; i >= 0; i--)
+            if (listView.isItemChecked(i)) count++;
+         return count;
+      }
```

(b) A Commit that reuse the code snippet

Fig. 1. (a) Source code snippet posted on StackOverflow, (b) Description of a commit that reuses the code snippet in WordPress Android.

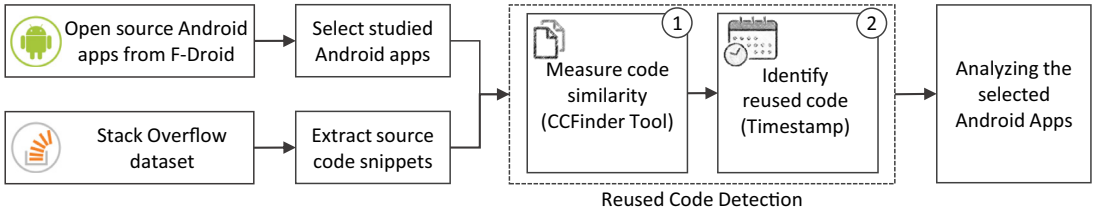


Fig. 2. Approach overview of our study.

Table 1
Selection process of StackOverflow posts.

Step	# Posts
All posts in the StackOverflow dataset	24,120,523
Posts tagged with Java	810,071
Posts tagged with Java and Android	106,861
Answer posts contain source code snippets	53,683
Code snippets with >= 30 lines	7458

Table 2
Selection process of the studied mobile apps.

Step	# Apps
Downloaded from F-Droid	1591
The ones we were able to run CCFinder on	1496
That have at least one clone from StackOverflow	377
That reused at least one case of code from StackOverflow	22

increase the scalability of the clone detection tool. In the end, our dataset contains 7458 posts. Table 1 summarizes how we arrived at this final number of posts.

Extracting Code Snippets from StackOverflow

Once we identified all of the StackOverflow posts that contained source code, we needed to extract the code snippet from the rest of the post. We applied a lightweight heuristic to identify the code in the StackOverflow posts. In StackOverflow, the body of a post is provided in HTML. Code elements in these posts are surrounded by the <code> tag. Therefore, we identify the code elements by searching for the <code> tags. Once we extracted the code elements, we manually checked the contents of the text between these <code> tags. We found that many posts do not only contain Java code. For example they may contain stack traces, XML code, plain text or URLs. To eliminate this non-Java code, we read the <code> tag contents and removed such code by applying regular expressions to filter out lines that start with special characters such as <, #,?,$,.,-, or a blank line. Moreover, even for the posts with Java code we eliminated all lines that begin with import statements. Performing the aforementioned preprocessing steps are necessary to further improve the matching between the StackOverflow code snippets and code from the mobile app.

3.2. Selecting mobile apps

In addition to obtaining the StackOverflow code snippets, we require the source code from mobile apps to study the code reuse. For our study, we resorted to the well-known F-Droid repository [14]. At the time of writing this paper, F-Droid included 1591 open source mobile apps. In addition to providing the APKs of each app, F-Droid provides a link to the source code of each app, which we used to extract the source code of the mobile apps.

Table 2 presents a summary of the steps, which we performed to arrive at our dataset of 22 mobile apps. For each of the 1591 apps, we examined the amount of reuse in the app exploiting an existing clone detection tool (step ① in Fig. 2, which is described later in this section). We were able to run the clone detection tool successfully on 1496 of the 1591 downloaded apps. Of the 1496 apps only 377 had one or more clones from StackOverflow code. Next, we filtered apps that had at least one instance of code reuse from StackOverflow. We make this check by comparing the date that the code segment was inserted in the project, using the commit date of the code with the date that the StackOverflow code was posted. If the commit date is *after* the date of the StackOverflow post, then we can conclude that the code is reused from StackOverflow (step ② in Fig. 2). This additional filter step further reduced our original dataset to 22 mobile apps.

Table 3
Descriptive statistics of the 22 mobile apps and the percentage of reused code from StackOverflow.

ID	App's Name	Category	#Commits	#Contributors	#LOC	% Reused
1	OsmAnd	Travel & Local	23,124	419	151,264	0.20
2	Open Explorer	Productivity	1669	10	130,565	0.07
3	WordPress	Social	11,467	54	69,760	0.19
4	AnkiDroid	Education	7463	110	45,088	0.08
5	Barcode Scanner	Tools	3152	77	42,514	0.38
6	ForPDA	Social	170	2	42,254	1.30
7	APG Encrypt	Communication	4366	68	41,564	0.70
8	Xabber Classic	Communication	1005	15	37,091	0.30
9	Smart Receipts Pro-	Finance	301	2	28,136	0.80
10	F-Droid	Tools	2550	55	21,039	0.48
11	FrostWire	Media & Video	3763	28	19,713	0.10
12	Andlytics Track	Shopping	1388	24	16,694	0.29
13	Open Training	Health & Fitness	501	6	10,264	1.36
14	BeTrains NMBS/SNCB	Transportation	209	7	9421	3.52
15	YASFA	Social	21	2	9128	1.57
16	Secrecy Secure file storage	Tools	155	1	6549	1.95
17	Tram Hunter	Travel & Local	260	5	5516	0.42
18	OpenLaw	Book & Reference	333	1	4037	0.37
19	OCR Test	Productivity	101	1	3910	5.70
20	OpenDocument Reader	Business	233	1	2996	0.87
21	blippex	Tools	18	3	2050	1.61
22	AnagramSolver	Word	46	2	745	0.94
					Average	1.06
					Median	0.59

To illustrate the diversity of our mobile app dataset, we present various app statistics in Table 3. As presented in Table 3, the apps belong to a number of different categories, that vary by the number of commits and contributors. Finally, the studied apps range in size from 745 to 151,264 lines of code (LOC).

3.3. Detection of reused code from StackOverflow in the mobile apps case-study

Once we created our StackOverflow code snippets corpus and the code from the mobile apps, our next step is to detect the reused code from StackOverflow within the mobile apps. We use the CCFinder[15] clone detection tool. CCfinder is a token based clone detection tool developed to detect Type-1 and Type-2 clones [15]. Three types of clones can be distinguished: Type-1 clones, which means that the two detected code snippets are identical. Type-2 clones, which means that the two source code snippets have the same structure except for variation in identifiers and literals, while Type-3 consider two code snippets to be similar even with further modification than Type-2 clones [16,17]. In our study, we detect similar code snippets utilizing Type-1 and Type-2 clones. Restricting our analysis to type-1 and type-2 clones a) reduces the number of potential false positives compared to type-3 clones, especially given that StackOverflow contains a large amount of code snippets, while b) providing an acceptable recall (compared to just using type-1 clones).

We execute CCFinder using its default configuration, i.e, the minimum length of the detected code clones is 50 tokens. It is important to note that although the code snippets extracted from StackOverflow needed to be a minimum of 30 lines, the overlap between any code from a mobile app and a StackOverflow code snippet has to be only 50 tokens, for the code fragment to be flagged as a clone. For example, we may have a case where only 5 lines (which have more than 50 tokens) from a mobile app appear in a StackOverflow snippet that is 30 lines long.

We selected CCFinder as our clone detection tool for several reasons. First, it detects type-1 and type-2 clones. Second, it is very efficient with respect to CPU and memory usage. Finally, it is freely available for research purposes. CCfinder returns clone groups with different sizes based on the number of matched tokens. In some cases, CCFinder may indicate multiple clones (of different sizes) in one code segment. In such cases, we take the largest clone as the match, since we need to detect the most similar code to the StackOverflow code snippet.

It is important to note that developers who reuse source code from StackOverflow occasionally preform obfuscation operations (e.g., renaming variable, adding and removing code) on the copied code. These modifications are sometimes done unintentionally to meet the system's new context and quality (e.g., programming style). On the other hand, some developers deliberately hide the reused code so that it becomes difficult to detect such reuse. Such deliberate hiding is done to reduce code licensing or ownership issues.

Once we obtain the code snippets that are reused from StackOverflow, we analyze the amount of reuse in the studied mobile apps, which we discuss next.

4. Preliminary analysis

Prior to delving into our research questions, we performed a preliminary analysis to quantify how much code reuse occurs from StackOverflow. Since recent work has shown that mobile developers often resort to StackOverflow [3,8], this investigation helps in quantifying how much (in terms of code reuse) StackOverflow is being used as a resource by mobile app developers.

To perform our analysis, we measure reuse in two complementary ways. First, we measure the percentage of StackOverflow posts that are reused in mobile apps. Second, we measure the percentage of code within mobile apps, which is reused from StackOverflow. We chose to use percentages instead of raw numbers since they allow us to easily compare the values.

Our analysis shows that from the 7458 posts in our StackOverflow code snippet corpus, 99 posts contain code snippets that were reused in the 22 studied mobile apps. This shows that approximately 1.33% of the StackOverflow posts are reused in the 22 studied mobile apps. It is important to note however, that in some cases, some posts in our corpus are reused more than once. As for the amount of a mobile app's code that is reused, Table 3 (column 7) shows the percentage of the app that is reused from StackOverflow. Table 3 shows that the OCR Test app (5.70%) has the highest amount of source code originating from StackOverflow, while the Open Explorer app (0.07%) has the smallest amount of reused

code. The average amount of code reused from StackOverflow for an app is 1.06% and the median is 0.59%.

Approximately 1.33% of the StackOverflow posts in our dataset are reused in the examined mobile apps. Moreover, on average 1.06% (0.59% median) of the examined mobile apps' source code is reused from StackOverflow.

5. Case study results

Thissection presents and discusses the results related to our research questions. For each research question, we present the motivation behind the question, the approach, our findings and their implications.

(RQ1) Why do mobile developers reuse code from StackOverflow?

Motivation: We saw earlier that code reuse does occur from StackOverflow. One of the first questions that comes to mind is *why* do mobile developers reuse code from StackOverflow? Answering this question will help us and the research community to better understand the types of activities and tasks which mobile app developers resort to StackOverflow for. In addition, our analysis will provide some insights into the type of reuse benefits mobile app developers received from StackOverflow. It is important to note here that our analysis is in the context of code reuse from StackOverflow for mobile apps and not the general use of StackOverflow as studied in prior work [3].

Approach: To answer this research question, we performed a qualitative analysis, where we manually examined the commit messages of the commits that introduced the StackOverflow code in each of the 22 mobile apps. We observed that the reused code was related to 140 different commits. We printed the message associated with each commit and examined all of them in turn. Five of the commits had commit messages that were unreadable (the message was corrupted), hence, we discarded them from our dataset. Our final dataset contained a total of 135 commit messages that introduced the StackOverflow code in the 22 selected mobile apps.

To determine the reasons why mobile developers reuse code from StackOverflow, we applied a manual analysis to discover and categorize the commits. Two graduate students (1 PhD student and 1 master student) separately open coded each commit message. Each student came up with their own categories by reading and analyzing the commit messages. After the participants completed their classifications, they met and discussed the commit messages that were not consistently classified (i.e., each member grouped these messages in different categories) to reach an agreement. The 135 commits were grouped into seven different groups, which were derived from the examined commit messages. Finally, we used Cohen's Kappa coefficient [18] to evaluate the level of agreement between the two coders. The Cohen's Kappa coefficient is a well-known statistical method that is used to evaluate the inter-rater agreement level for categorical scales. The resulting coefficient is scaled to range between -1.0 and +1.0, where a negative value means poorer than chance agreement, zero indicates exactly chance agreement, and a positive value indicates better than chance agreement.

Findings: As a result of our manual classification process, we ended up with seven different categories that the commit messages were grouped into. We observe that mobile developers tend to reuse code from StackOverflow for different reasons such as: using API, fixing bugs, testing, refactoring, adding new features and enhancing existing code. In some cases, there were commit messages that were not descriptive enough to allow for a clear classification, e.g., the message would simply contain a sequence of digits or only contain one word such as "initial", however, this was a small percentage of the examined commits. Table 4 presents the percentage of commits for each category. We observe that enhancing existing features and adding new features are the two most common reasons for code reuse from StackOverflow in mobile apps, accounting for approximately 57% of the examined commits. Additionally, we found that Cohen's Kappa coefficient shows the level of agreement between the two coders to be +0.82, which is considered to be an excellent agreement [19].

Table 4
Reuse categories based on coding of commit messages.

Category	% of Commits
Enhancing existing code	33.33%
Adding new features	23.70%
Refactoring	12.59%
API usage	11.11%
Fixing bugs	10.37%
Test	0.74%
Other	8.14%

Potential Implications: Based on our findings, there are a number of implications for our RQ1. First, we believe that *the StackOverflow community* will know what mobile developers are using their posts for, so they can provide better support for such activities (e.g., in addition to sharing code, one can point to possible documentation since developers are mostly using the code to implement new features). Other more drastic measures can involve asking users or contributors of the code to provide quality assurance mechanisms (e.g., tests or code reviews) for snippets that are deemed to have a higher probability of reuse. Our findings are also useful for *the mobile developer community* since they will know what tasks other developers are reusing code from StackOverflow for, e.g., they will know that StackOverflow may contain relevant resources to help with refactoring. Finally, *the research community* will know in what context mobile app developers reuse code from StackOverflow, so they can develop techniques and tools to assist with such tasks, e.g., use StackOverflow to help with testing, though such cases are rare in our dataset.

Mobile app developers reuse code from StackOverflow to use APIs, fix bugs, conduct testing, refactor existing code, add new features and enhance existing code. The most common reason for reuse from StackOverflow is the enhancement of existing code.

(RQ2) When in a mobile app's lifetime do developers reuse code from StackOverflow?

Motivation: After examining the various reasons for source code reuse from StackOverflow, we would like to know when in the project's lifetime mobile app developers tend to reuse code the most. Answering this question helps us better understand at what stage of development mobile developers need the most help and resources. It also tells whether the impact of reuse can only be expected late in the mobile app (in case we find most reuse happens later on) or throughout the project's lifetime.

Approach: To address this research question, we first compute the age of each app in number of days. We then classify the apps in terms of their maturity in terms of days since their first commit. This classification enables us to perform a fair comparison since

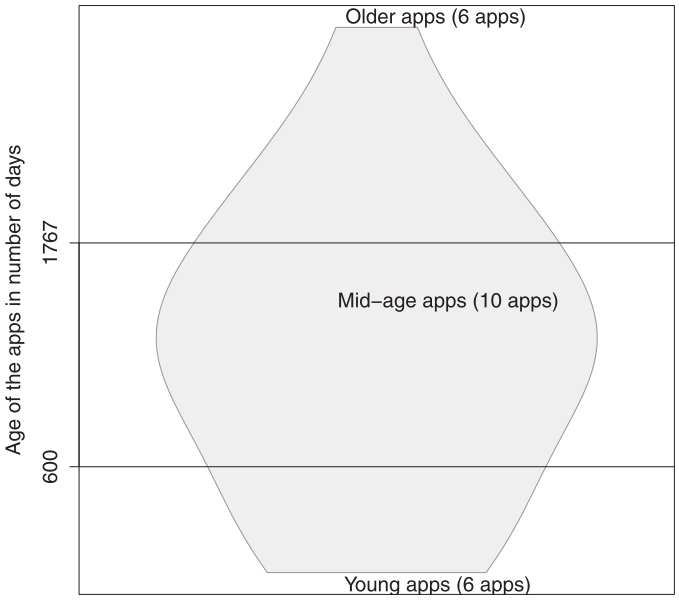


Fig. 3. Distribution of the app's age of the 22 examined mobile apps.

different projects will have different ages (i.e., lifetime). The distribution of ages of the 22 apps is shown in Fig. 3. We use the first and third quantiles to divide the apps into three different maturity groups. As shown in Fig. 3, the first quantile is at 600 days and the third quantile is at 1767 days. Based on this distribution, we divide the apps into 1) young apps (*app's age* $<$600 *days*), 2) mid-age apps (600 *days* $=<$ *app's age* $=<$ 1, 767 *days*), and 3) older apps (*app's age* $>$ 1, 767 *days*). Based on this division, we end up having 6 young apps, 10 mid-age apps and 6 older apps.

After classifying the apps into three groups, i.e., young, mid-age and older apps, we counted the percentage of commits that reuse code in each app. To gain a finer grained view of the reuse, we measured the percentage of reused commits for each app per quartile (i.e., in the first, second, third or fourth quartile). Such a fine grained view can tell us, for example, whether most of the reuse for mid-age apps occurs early (e.g., first quartile) or later on (e.g., fourth quartile) in their lifetime.

We measure the reuse as the percentage (rather than raw number) of commits since the raw number of commits in each app (and each quartile) can vary. To ensure that we only count relevant commits, we remove all merge commits from our calculation of the total commits.

Findings: Fig. 4 shows bean plots (with superimposed boxplots) of the percentage of the code reuse from StackOverflow for the three mobile app groups. Bean plots are useful in presenting the distribution of data, whereas the superimposed box plots highlight key statistics. Fig. 4a shows that for younger apps, most code reuse from StackOverflow occurs in the middle of the apps' lifetime, i.e., second and third quartiles, yet this difference is not statistically significant. To determine whether there are statistically significant differences between the values in the different quartiles, we perform a one-way analysis of variance (ANOVA), and we could not observe a statistically significant difference. Fig. 4b shows that for mid-age apps, most reuse occurred late in the mobile apps' lifetime (i.e., fourth quartile). ANOVA showed that there is a statistically significant difference (*p-value* is $<$0.05) between the median of the fourth quartile and the medians of the other quartiles. For the older apps, we observe that, once again, most reuse from StackOverflow happened late in the mobile app lifetime, i.e., the third and fourth quartiles. Also, the use of ANOVA shows that the observations for the third and fourth quartiles are statistically significant, with *p-value* $<$0.05. Our findings indicate that although developers tend to reuse code for feature addition and enhancement, this reuse (feature addition and enhancement) varies based on the age of the app.

Potential Implications: There are a number of implications of our findings in RQ2. For *the StackOverflow community*, our findings show that it is not just young or immature mobile apps that reuse code from StackOverflow, rather even more mature or older mobile apps tend to reuse code from StackOverflow. Hence, proper mechanisms can be developed to tailor results that are returned to users based on the age of their mobile app. Alternatively, StackOverflow could add a mechanism for explicitly rating the source code snippets that have been reused in actual mobile apps. For example, developers of younger apps are looking more for questions/code related to initial development tasks, whereas developers of older apps are more concerned with fixing, refactoring, or maintenance issues. For *the mobile developer community*, our findings show that indeed, other developers resort to StackOverflow even at later stages in the development of their projects. We believe that developers should link to StackOverflow posts that they used to help reach their final coded solutions. For *the research community*, our findings can be used to motivate the development of techniques that support maintenance or late-stage development with resources from StackOverflow.

> *Mobile app developers tend to reuse code from StackOverflow at later stages of the project for mid-age and older mobile apps.*

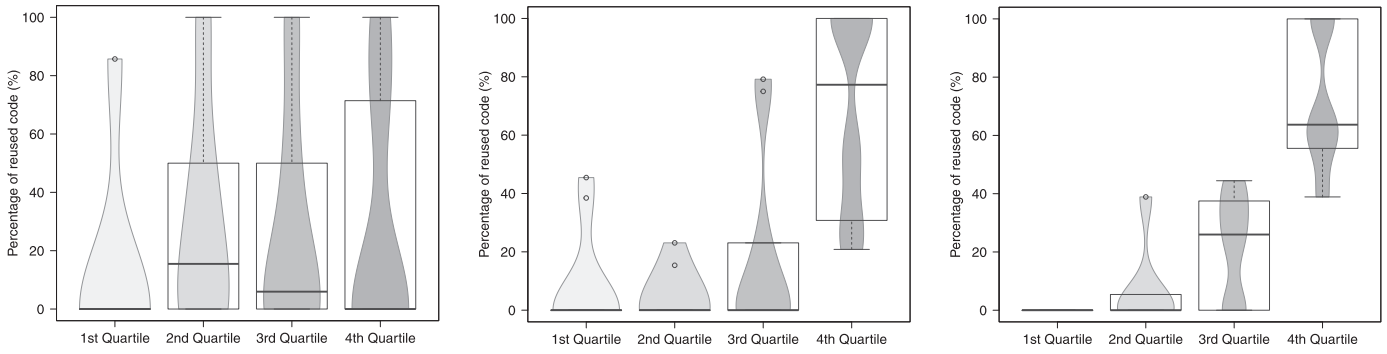
(RQ3) Who reuses code from StackOverflow?

Motivation: Prior work has indicated that reusing code is negatively perceived by developers and associated with less experienced developers [20]. Hence, we wanted to empirically examine who reuses StackOverflow code amongst mobile developers, e.g., is it more common among less experienced developers that are looking for quick solutions or is it the experienced developers who resort to code reuse from StackOverflow?

Approach: To answer this research question, we first measure the developer experience within the mobile app. Similar to prior work [21,22], we use the developer activity, measured using the number of previous commits from the start of the project to the time of code reuse, to determine a developer's experience level. We use this method "previous number of changes" because 1) the previous number of changes done by the developer represents a meaningful proxy of a developer's experience that can be accurately measured and 2) previous work uses the prior number of commits as a measure of a developer's experience [21–23].

We observed that some of the developers commit from different emails and with a variety of different names. Therefore, we manually examined the names and email addresses of all developers in our dataset and merged similar identities. In total, we found 37 unique developers who reused code from StackOverflow in the 22 examined mobile apps. Once again, we normalized the experience of the developers and present it as a percentage of the total number of commits in the project (i.e., a developer's experience in a project is measured as $\frac{\#\ commits\ by\ the\ developer}{total\ \ commits} \times\ 100$).

Findings: At first, we compare the experience of all mobile app developers who reuse code from StackOverflow against developers who do not reuse code from StackOverflow for the 22 studied mobile apps. Fig. 5 compares the percentage of developers' experiences for both developers who reuse StackOverflow code and developers who do not reuse StackOverflow code. Our study shows



(a) Young mobile apps (b) Mid-age mobile apps (c) Older mobile apps

Fig. 4. Percentage of reused StackOverflow code per quartile in apps divided based on the age.

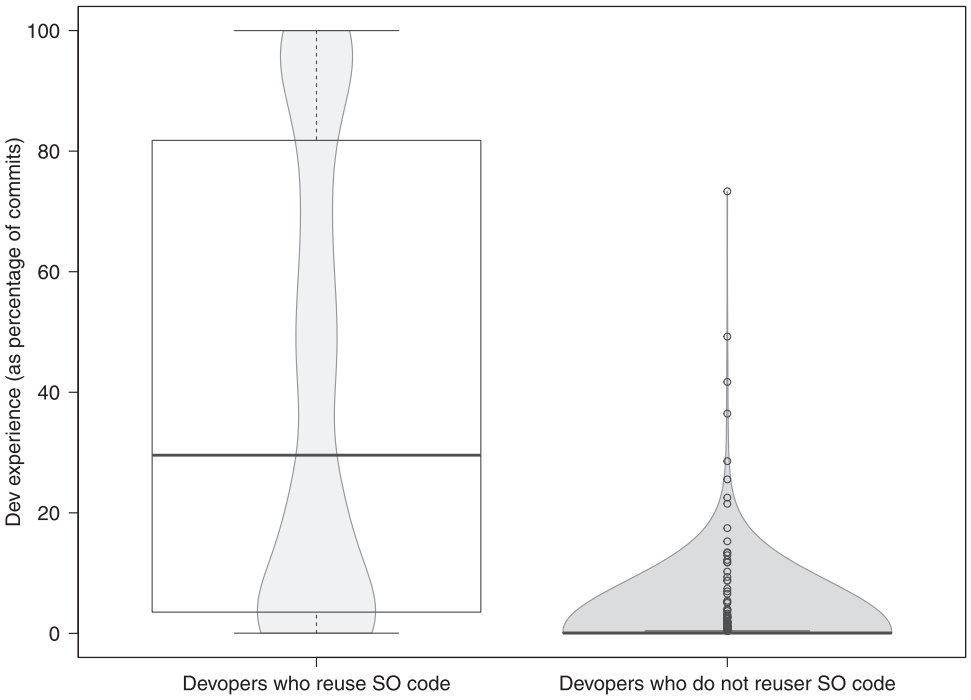


Fig. 5. Distribution of the developers' experience for developers who reuse code from StackOverflow against the rest of developers in the 22 studied mobile apps.

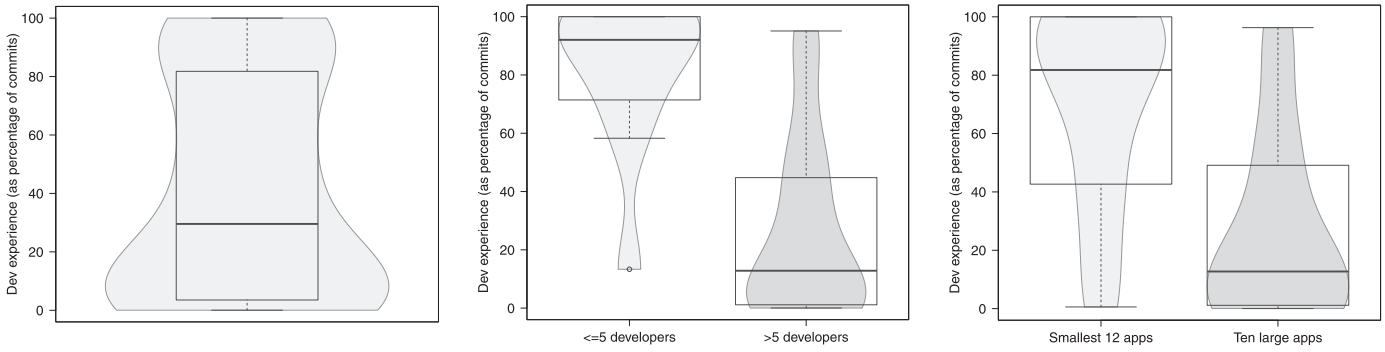
that StackOverflow code reuse is mainly performed by more experienced developers (median equal to 29.56%) than the rest of the developers (median 0.04%) in our 22 studied mobile apps. We also find that the difference between the percentage of developers' experiences in the two groups is statistically significant, with a *p-value* < 0.05.

Subsequently, we examined in more detail the experience of all mobile app developers (in terms of their commit activity) who reuse code from StackOverflow. Fig. 6a shows the results of our analysis when all apps are grouped together. The figure shows that on median, developers who commit 29.56% of the total commits are the ones who reuse code from StackOverflow. The box in the box plot also shows wide variation, which in essence makes it difficult to observe any type of trend.

Thus, we decided to perform the in depth analysis based on both team size and LOC of the apps since they are two factors that can impact the app, i.e., apps developed by smaller teams may follow different development processes compared to apps that are developed by larger teams and larger apps may do more than apps that are smaller in size, hence this was a clear and intuitive way to divide the apps.

We then repeated the same analysis for apps that are developed by small teams ($<= 5$ developers) and apps developed by larger teams (> 5 developers). Fig. 6b shows a clear difference when we do this division. From Fig. 6b we observe that in apps developed by smaller teams, more experienced developers (median 92.05%) tend to reuse code from StackOverflow, whereas in apps developed by larger teams, it is the less experienced developers (median 12.83%) who reuse code from StackOverflow. We also perform the Mann-Whitney test to identify if the observation is statistically significant, which confirms that the difference is statistically significant with the *p-value* being < 0.05.

We next considered the size of the apps for our analysis in terms of lines of code. We divided the apps into the top 10 largest and the remaining 12 apps. Fig. 6c shows that for the 10 largest apps, the less experienced (median 12.73%) mobile app developers tend to reuse code from StackOverflow. For the smallest 12 apps, the results shows that more experienced (median 81.78%) developers reuse code from StackOverflow. Finally, we utilize the Mann-Whitney test to identify if this difference is statistically significant. We found that the *p-value* is < 0.05, which confirms that the difference is statistically significant.



(a) All mobile apps (b) Apps divided based on No. of developers (c) Apps divided based on size (LOC)

Fig. 6. Distribution of the developers' experience for developers that reuse code from StackOverflow.

We also experimented with other thresholds. We first computed the number of developers of each app as a threshold to divide the mobile apps in our dataset into two groups. We use the median instead of mean because the test for normality shows that the data is not normally distributed (Shapiroâ;;Wilk test < 0.05). Further, we divided the apps based on the median (median $=$ 6.5, around 7). We found that again in apps developed by smaller teams (i.e., $=<$ 7 developers), more experienced developers (median 91.45%) tend to reuse code from StackOverflow, while in apps developed by larger teams, it is less experience developers (median 12.63%) who reuse code from StackOverflow. Also, the Mann-Whitney test shows that the difference is statistically significant with the p-value being < 0.05.

We also divided the mobile apps in our dataset based on the median of apps size (LOC) into small and large apps. We use median (median $=$ 18200) because, once again, the test for normality shows that the data is not normally distributed (Shapiroâ;;Wilk test < 0.05). The results show that in small apps developers with more experience tend to reuse source code from StackOverflow with median equal to 84.79%, while in large apps developers with less experience reuse code form Stack Overflow (median $=$ 12.83%). The Mann-Whitney test shows that the difference is statistically significant with the p-value being < 0.05. Based on these experiments, we found the same result as dividing the apps based on the median size of the apps.

Potential Implications: There are a number of implications from our findings in RQ3. For the *StackOverflow community*, our findings can motivate the need to develop techniques that analyze and attach a "safety index" to code snippets since our results provide evidence that for larger teams/apps, less experienced developers tend to reuse code from StackOverflow. For the *mobile developer community*, our findings show that in larger teams and apps, code developed by junior or less experienced developers needs to be more closely reviewed since it may be code that is reused. Such reuse can also have licensing implications, although this topic is beyond the scope of this paper. For the *research community*, our results can serve as motivation for the need to perform more fine grained studies on reuse by different types of developers. Additionally, our findings can help shed light on code ownership (since we see who reuses code is also related to the size of the team or project the developer works with), a topic that has received increasing attention lately.

More experienced mobile app developers reuse StackOverflow code in smaller teams/apps, whereas less experienced developers reuse StackOverflow code in larger teams/apps in our dataset.

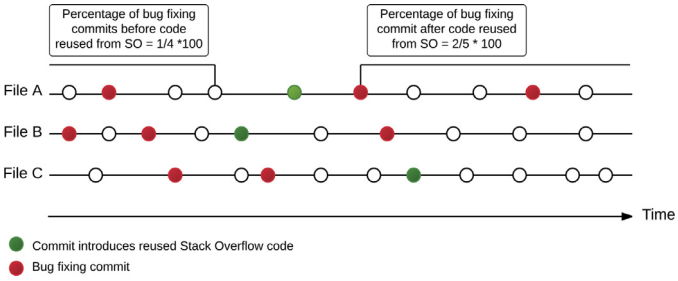


Fig. 7. The approach to compute the percentage of bug fixing commits *Before* and *After* reuse per file.

6. Does code reuse from StackOverflow impact the quality of mobile apps?

Thus far, we have investigated the *how much, why, when* and *who* questions related to reusing code from StackOverflow in mobile apps. Our findings showed that the amount of reused StackOverflow code varies for different mobile apps, that feature additions and enhancements are the main reasons for code reuse, that mid-age and older apps reuse StackOverflow later in their lifetime, and that more experienced developers reuse code in smaller teams/apps, while less experienced developers reuse code in larger teams/apps.

Additionally, we wanted to examine the implications of code reuse from StackOverflow on the quality of mobile apps. Hence, we examine the bug fixing commits of files that reuse code from StackOverflow *before* and *after* the reuse occurred. For this analysis as shown in Fig. 7, we marked all files that contain reused StackOverflow code in all of the 22 studied mobile apps. Then, for each file we retrieve all the commits that ever touched the file. We identified the commit that introduces the reused code from StackOverflow and divided the commits into two groups: commits that occurred *before* and commits that occurred *after* the introduction of the reused code. We used heuristics to classify each commit as a bug fixing or non-bug fixing commit. Similar to prior work [24,25], we use a set of keywords to identify bug fixing commits. A commit is identified as a bug fixing commit if its commit message contains one of the following keywords "fix", "bug", "defect", "patch", "error". We then compute the ratio of bug fixing commits by dividing the number of fixing commits by the total number of commits for each file *before* and *after* the introduction of the reused code. Finally, we compare the percentage of bug fixing commits in the two periods, *before* and *after* reusing StackOverflow code. We compare the percentage of bug fixing commits instead of the raw numbers since StackOverflow code could be introduced at different times,

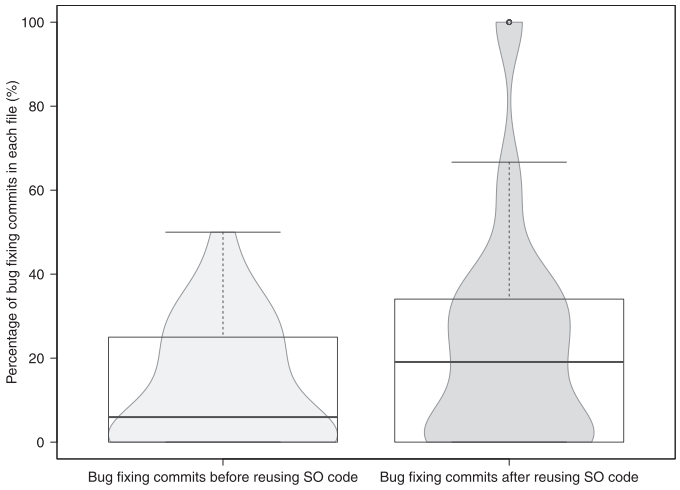


Fig. 8. The Percentage of bug fixing commits *Before* and *After* reuse per file.

i.e., we may not have the same total number of changes before and after the introducing StackOverflow code. Doing this will show if a higher percentage of bugs exists after code is reused from StackOverflow.

Fig. 8 shows the distribution of percentage of bug fixing for all files. We observed that the median percentage of bug fixing *before* reusing the StackOverflow code is 5.96% and the max is 50%. On the other hand, the right box plot represents the percentage of bug fixing *after* reusing Stack Overflow code. We see that the percentage of bug fixing commits is higher after the code reuse from StackOverflow (median equal to 19.09%). To determine if this difference is statistically significant, we perform a Mann-Whitney test, which confirmed that the difference is statistically significant with p-value < 0.05.

In addition, we computed the effect size of the difference using Cliff's Delta (d), which is a non-parametric effect size measure for ordinal data. Cliff's d ranges in the interval [-1, 1] and is considered small for $d < 0.33$ (positive as well as negative values), medium for $0.33 \leq d < 0.474$, and large for $d \geq 0.474$. The result shows that the difference has a small effect size (Cliff's $d = 0.225$) when comparing the percentage of bug fixing commit *before* reusing the StackOverflow code and the percentage of bug fixing commit after the reuse of source code from StackOverflow.

Although this is not a comprehensive study on the impacts of code reuse in mobile apps, we see our aforementioned finding as preliminary evidence that reuse from StackOverflow may have a negative impact on the quality of mobile apps.

7. Related work

Work related to our research can be divided into three categories: research on the origin of source code and code reuse, work related to the use of StackOverflow, and work related to code reuse in mobile apps.

7.1. Origin of source code and code reuse

Inoue et al. [2] developed a prototype called Ichi Tracker to explore the evolution of a code fragment utilizing online code search engines and code clone detection techniques. The tool accepts a code fragment as input and returns related files containing query code. Using the Ichi Tracker, developers can identify the origin of a source code fragment or a modified version of the code fragment, including potential license violations. German et al. [11] examined source code migration across three different systems (Linux, FreeBSD and OpenBSD) and its legal implications. They tracked reused code fragment using clone detection methods. Their result showed that code migration did occur between these systems. Additionally, the copying tended to be performed without violating the license terms. Davies et al. [26] proposed a signature-based matching technique to determine the origin of code entities. They found that their technique can be utilized to identify security bugs in the reused libraries. Kawamitsu et al. [27] proposed a technique to automatically detect source code reuse between two software repositories at the file level. It is based on measuring the similarity between two source files and using the commit time to identify the original source file revision. They found that in some instances developers did not record the version of the reused file.

Our work differs from this existing research in several aspects. First, the main focus of our work is on reuse of source code from StackOverflow in the context of mobile apps. Secondly, while our work is similar to some existing work (e.g., [11]), in that we also detect source code reuse using a clone detection technique, our focus is analyzing why, when and who reuses code from StackOverflow. This is in contrast to existing work, where the focus had been on detecting the origin of source code, license violations and code migration.

7.2. Work related to the use of StackOverflow

Barua et al. [8] proposed a semi-automatic approach to study general topics discussed on StackOverflow by developers. They found that web and mobile development are the most popular topics. Rosen and Shihab [3] used StackOverflow to determine what mobile developers on StackOverflow ask about. They found that questions posted on StackOverflow cover almost all issues related to the development of mobile apps, and that app distribution and user interface questions are the most viewed. We consider the prior results of these studies as evidence that StackOverflow is a very popular source of programming-related knowledge and source code, which served as a motivation for our work. In our study however, we focus on the reuse of source code from StackOverflow in mobile apps. More specifically we propose an approach to identify reused code from StackOverflow and answer a number of research questions such as, why mobile app developers reuse code from StackOverflow, when in a mobile apps lifetime developers reuse code from StackOverflow, what is the experience of these developers, and the potential impact of reusing code from StackOverflow on software quality. To the best of our knowledge, our study is the first to empirically study code reuse from StackOverflow in mobile apps.

Several other work has attempted to recommend code snippets from StackOverflow. Cordeiro et al. [28] indexed the StackOverflow data dump to retrieve associated discussion. They extract keywords from exception traces in Eclipse to automatically suggest Q&A threads from StackOverflow to developers. Ponzanelli et al. [9] developed a tool called Prompter that recommends related code from StackOverflow. Prompter generates a search query from the developers' programming context in the IDE and searches StackOverflow to recommend code snippets. Rahman et al. [29], proposed SurfClipse that provides immediate assistance to developers when they encounter runtime errors or exceptions. It exploits the code terms to search in three search engines and StackOverflow, and shows the collected results in the Eclipse IDE. Wang et al. [30], devised an approach to build a bidirectional link between the Android issue tracker and StackOverflow discussions to facilitate the knowledge sharing between two separated communities. They exploit the semantic similarity with temporal-locality to effectively establish the link. Once such a link is established, it allows contributors (developers and users) in the two communities to gain the benefit of knowledge sharing. Our prior work exam-

ined why developers use StackOverflow and found, amongst other findings, that developers do reuse code from StackOverflow [31].

In contrast to existing research that mainly focused on recommending code snippets form StackOverflow to assist developers, our work's main goal is to perform an exploratory study on code that is reused from StackOverflow in mobile apps.

7.3. Work related to mobile apps

Several studies on mobile software development have been published in recent years. Syer et al. [32] compared the source code of BlackBerry and Android apps along three aspects, source code, code dependencies and code churn. They found that BlackBerry apps are larger and rely more on third party libraries, whereas Android apps have less number of files and rely mostly on the Android platform. Ruiz et al. [33] compared the extent of code reuse in the different categories of Android apps. They observed that around 23% of the classes inherit from one of the base classes in the Android API and 27% of the classes inherit from a domain specific base class. Furthermore, they found that 217 mobile apps are completely reused by another mobile app. Minelli and Lanza [34] proposed a software analytics platform called SAMOA that has been utilized to analyze 20 Android apps. Their analysis showed that mobile apps intensely depend on the usage of external APIs. Chen et al. [35] developed an approach, based on geometry characteristics, that can detect clones between mobile apps. Our work differs from this related work in that we focus on code reuse from StackOverflow and not across mobile apps or from APIs.

8. Threats to validity

Threats to internal validity concern confounding factors that could have influenced our study results. First, to identify the reused code from StackOverflow, we use the CCFinder clone detection tool. Hence, we are limited by the accuracy of CCFinder in finding Type-1 and Type-2 clones. In some cases, we may have missed instances of code reuse if CCFinder did not flag the code as a clone. To help alleviate this issue, we manually investigated some of the clones and in all cases the flagged clones were reused code from StackOverflow. In addition, we use CCfinder with the default configuration parameters. Changing these parameters may lead to different results. When we identify reused code, we ensure that the date of the StackOverflow post is prior to the commit date. In certain cases, the code in the project may not have actually originated from StackOverflow, if for example the developer takes a long time to commit. Furthermore, we found there are some cases where StackOverflow posts are reused more than once that developers may copy the source code snippets between mobile apps.

To determine why developers reuse code form StackOverflow, we manually analyzed the commits. Like any human activity, the categorization may be prone to human error or bias. To alleviate this threat, we had two students separately code the commits messages and come to an agreement on any discrepancies. We also computed Cohen's Kappa to evaluate the inter-rater agreements, which showed excellent inter-rater agreement (Cohen's Kappa value of +0.82). In addition, we use the commit message that introduced the code as an indicator of the reuse. However, a commit may contain the StackOverflow code combined with other code. In this analysis, we consider a commit as a single unit of change with the reused code being part of the complete unit, hence, the reason should be the same for all the modified code. Similar to prior work, e.g., [21,23], we use the number of previous commits to measure experience. In some cases, the number of previous commits may not be representative of the actual experience of a developer. Furthermore, given that we manually identified the unique developers by comparing names and email addresses of all developers, it is possible that we potentially wrongly identified a developer as unique. The analysis in RQ2 is based on the size of the apps and number of developers, which may not present all the development features of apps.

Threats to external validity concern the possibility that our results may not be generalizable. In this study, we focus on StackOverflow, which is one of many Q&A websites, hence, our results may not generalize to reuse from other Q&A websites. In addition, we examined 22 Android Apps in which code reuse occurred, hence, our findings may not generalize to other apps, especially apps that are not open source. Finally, we studied source code reuse from StackOverflow in the context of mobile apps, which may not be generalizable to other application domains.

9. Conclusion and future work

In this paper, we investigated the reuse of code from StackOverflow in mobile apps. We conducted an exploratory study using 22 Android apps. We found that the amount of reused StackOverflow code varies for different mobile apps, that feature additions and enhancements are the main reasons for code reuse from StackOverflow, that mid-age and older apps reuse StackOverflow code later in the lifetime and that more experienced developers reuse code in smaller teams/apps, while less experienced developers reuse code in larger teams/apps. We also examined the potential implications of code reuse and found that code reuse is related to higher potential of bug fixing. In the future, we plan to expand our study to include other types of Q&A websites, as well as more mobile apps.

The results from our studies not only provide some valuable insights into the potential reuse of code from StackOverflow in mobile apps, but also provide insights into challenges for making code reuse from Q&A websites such as StackOverflow part of current mobile app development processes and best practices.

In addition to our main findings, we believe that the technique of detecting code reuse from StackOverflow developed as part of this work can have an impact and enables the investigation of other phenomena in mobile apps and software engineering in general. For example, our approach can help enable:

Bi-directional traceability: Currently, knowledge, including code snippets, from Q&A websites and mobile project code repositories remains in information silos, preventing feedback loops across these knowledge resources. As a result, no mechanism is in place to allow for notifications or monitoring changes, which might have an effect on either source. For example, in response to the original question posted on StackOverflow, an improved solution is posted or a problem (e.g., vulnerability) might have been discovered. However, establishing and maintaining such links will require an adequate tool support. Thus, our approach may help improve traceability of StackOverflow code reused in mobile apps.

Detection of licensing violations: An important aspect that mobile app developers should pay attention to when reusing code from StackOverflow is the issue of potential copyright violations. Given that the origin of the code posted on Q&A websites is often unknown and different Q&A websites take advantage of a variety of open source licenses, avoid potential license violations when reusing code snippets. For example, the StackOverflow website operates under a general CC-BY-SA open source license, allowing code snippets posted on StackOverflow to be reused and adapted. Mobile app developers reusing code are therefore required to provide appropriate credit to the source and distribute their code under the same license to avoid license violations.

Improved content and quality rating: Information of actual code reuse from the StackOverflow website source code can be used to improve the rating mechanism on StackOverflow. The frequency and context of code reuse from code snippets originat-

ing from StackOverflow can be used to improve and establish more meaningful rating and tagging mechanisms for StackOverflow posts, depending on actual real world usage scenarios.

In the future we plan to apply our approach to investigate its applicability in the aforementioned topics. We also plan to extend our work to consider reuse in other contexts.

References

[1] W. Lim, Effects of reuse on quality, productivity, and economics, IEEE Software 11 (5) (1994) 23–30.

[2] K. Inoue, Y. Sasaki, P. Xia, Y. Manabe, Where does this code come from and where does it go? - integrated code history tracker for open source systems -, in: Proceedings of the 34th International Conference on Software Engineering (ICSE), 2012, pp. 331–341.

[3] C. Rosen, E. Shihab, What are mobile developers asking about? a large scale study using stack overflow, Empirical Software Engineering (EMSE) 21 (3) (2016) 1192–1223.

[4] C. Sadowski, K.T. Stolee, S. Elbaum, How developers search for code: a case study, in: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (FSE), 2015, pp. 191–201.

[5] J. Gui, S. Mcilroy, M. Nagappan, W.G.J. Halfond, Truth in advertising: The hidden cost of mobile ads for software developers, in: Proceedings of the 37th International Conference on Software Engineering (ICSE), 2015, pp. 100–110.

[6] M.D. Syer, M. Nagappan, A.E. Hassan, B. Adams, Revisiting prior empirical findings for mobile apps: An empirical case study on the 15 most popular open–source android apps, in: Proceedings of the 13th Conference of the Center for Advanced Studies on Collaborative Research (CASCON), 2013, pp. 283–297.

[7] I.J. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, A.E. Hassan, A large-Scale empirical study on software reuse in mobile apps, IEEE Software 31 (2) (2014) 78–86.

[8] A. Barua, S.W. Thomas, A.E. Hassan, What are developers talking about? an analysis of topics and trends in stack overflow, Empirical Software Engineering (EMSE) 19 (3) (2012) 619–654.

[9] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, M. Lanza, Mining StackOverflow to turn the IDE into a self-confident programming prompter, in: Proceedings of the 11th Working Conference on Mining Software Repositories (MSR), 2014, pp. 102–111.

[10] S.M. Nasehi, J. Sillito, F. Maurer, C. Burns, What makes a good code example?: A study of programming Q&A in StackOverflow, in: Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM), 2012, pp. 25–34.

[11] D.M. German, M. Di Penta, Y.-G. Gueheneuc, G. Antoniol, Code siblings: Technical and legal implications of copying code between applications, in: Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories (MSR), 2009, pp. 81–90.

[12] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, M. Di Penta, R. Oliveto, D. Poshyvanyk, Api change and fault proneness: a threat to the success of android apps, in: Proceedings of the 9th joint meeting on foundations of software engineering (ESEC/FSE), 2013, pp. 477–487.

[13] F. Sarro, A.A. Al-Subaihin, M. Harman, Y. Jia, W. Martin, Y. Zhang, Feature lifecycles as they spread, migrate, remain, and die in app stores, in: Proceedings of IEEE 23rd International Requirements Engineering Conference (RE), 2015, pp. 76–85.

[14] F-Droid, Free and open source android app repository, 2015, [Online; accessed 2015-08-04].

[15] T. Kamiya, S. Kusumoto, K. Inoue, CCFinder: A multilinguistic token-based code clone detection system for large scale source code, IEEE Transactions on Software Engineering (TSE) 28 (7) (2002) 654–670.

[16] R. Koschke, Survey of Research on Software Clones, in: R. Koschke, E. Merlo, A. Walenstein (Eds.), Duplication, Redundancy, and Similarity in Software, number 06301 in Dagstuhl Seminar Proceedings, 2007.

[17] C.K. Roy, J.R. Cordy, A survey on software clone detection research, Technical Report, Technical Report 541, Queen's University at Kingston, 2007.

[18] J. Cohen, A coefficient of agreement for nominal scale, Educ Psychol Meas 20 (1960) 37–46.

[19] J. Fleiss, The measurement of interrater agreement, Statistics methods for rates and proportions (1981) 212–236.

[20] O. Barzilay, C. Urquhart, Understanding reuse of software examples: a case study of prejudice in a community of practice, Information and Software Technology (IST) 56 (12) (2014) 1613–1628.

[21] E. Shihab, A.E. Hassan, B. Adams, Z.M. Jiang, An industrial study on the risk of software changes, in: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE), 2012, pp. 62:1–62:11.

[22] A. Mockus, D.M. Weiss, Predicting risk of software changes, Bell Labs Tech J 5 (2) (2000) 169–180, doi:10.1002/bltj.2229.

[23] C. Bird, N. Nagappan, B. Murphy, H. Gall, P. Devanbu, Don't touch my code!: examining the effects of ownership on software quality, in: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering (ESEC/FSE), 2011, pp. 4–14.

[24] A. Mockus, L.G. Votta, Identifying reasons for software changes using historic databases, in: Proceedings of the 16th International Conference on Software Maintenance (ICSM), 2000, pp. 120–130.

[25] J. Eyolfson, L. Tan, P. Lam, Do time of day and developer experience affect commit bugginess? in: Proceedings of the 8th Working Conference on Mining Software Repositories (MSR), 2011, pp. 153–162.

[26] J. Davies, D.M. German, M.W. Godfrey, A. Hindle, Software bertillonage: finding the provenance of an entity, in: Proceedings of the 8th working Conference on Mining Software Repositories (MSR), 2011, pp. 183–192.

[27] N. Kawamitsu, T. Ishio, T. Kanda, R.G. Kula, C. De Roover, K. Inoue, Identifying Source Code Reuse across Repositories Using LCS-Based Source Code Similarity, in: Proceedings of the 14th International Working Conference on Source Code Analysis and Manipulation (SCAM), 2014, pp. 305–314.

[28] J. Cordeiro, B. Antunes, P. Gomes, Context-based recommendation to support problem solving in software development, in: Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering (RSSE), 2012, pp. 85–89.

[29] M.M. Rahman, S. Yeasmin, C.K. Roy, Towards a context-aware ide-based meta search engine for recommendation about programming errors and exceptions, in: Proceedings of 21th IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE), 2014, pp. 194–203.

[30] T. Wang, G. Yin, H. Wang, C. Yang, P. Zou, Proceeding of the automatic knowledge sharing across communities: A case study on android issue tracker and stack overflow, in: Proceedings of the 9th IEEE Symposium on Service-Oriented System Engineering (SOSE), 2015, pp. 107–116.

[31] R. Abdalkareem, E. Shihab, J. Rilling, What do developers use the crowd for? a study using stack overflow, IEEE Software 34 (2) (2017) 53–60.

[32] M.D. Syer, B. Adams, Y. Zou, A.E. Hassan, Exploring the development of micro-apps: A case study on the blackberry and android platforms, in: Proceedings of 11th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM), 2011, pp. 55–64.

[33] I.J.M. Ruiz, M. Nagappan, B. Adams, A.E. Hassan, Understanding reuse in the android market, in: Proceedings of IEEE 20th International Conference on Program Comprehension (ICPC), 2012, pp. 113–122.

[34] R. Minelli, M. Lanza, Software analytics for mobile applications–insights & lessons learned, in: Proceedings of 17th European Conference on Software Maintenance and Reengineering (CSMR), 2013, pp. 144–153.

[35] K. Chen, P. Liu, Y. Zhang, Achieving accuracy and scalability simultaneously in detecting application clones on android markets, in: Proceedings of the 36th International Conference on Software Engineering (ICSE), 2014, pp. 175–186.