2018 25th Australasian Software Engineering Conference (ASWEC)

Code Reuse in Stack Overflow and Popular Open Source Java Projects

Adriaan Lotter
Department of Information Science
University of Otago
Dunedin, New Zealand
adriaan.lotter@otago.ac.nz

Sherlock A. Licorish
Department of Information Science
University of Otago
Dunedin, New Zealand
sherlock.licorish@otago.ac.nz

Bastin Tony Roy Savarimuthu
Department of Information Science
University of Otago
Dunedin, New Zealand
tony.savarimuthu@otago.ac.nz

Sarah Meldrum
Department of Information Science
University of Otago
Dunedin, New Zealand
sarah-meldrum@outlook.com

***Abstract*— Solutions provided in Question and Answer (Q&A) websites such as Stack Overflow are regularly used in Open Source Software (OSS). However, many developers are unaware that both Stack Overflow and OSS are governed by licenses. Hence, developers reusing code from Stack Overflow for their OSS projects may violate licensing agreements if their attributions are not correct. Additionally, if code migrates from one OSS through Stack Overflow to another OSS, then complex licensing issues are likely to exist. Such forms of software reuse also have implications for future software maintenance, particularly where developers have poor understanding of copied code. This paper investigates code reuse between these two platforms (i.e., Stack Overflow and OSS), with the aim of providing insights into this issue. This study mined 151,946 Java code snippets from Stack Overflow, 16,617 Java files from 12 of the top weekly listed projects on SourceForge and GitHub, and 39,616 Java files from the top 20 most popular Java projects on SourceForge. Our analyses were aimed at finding the number of clones (indicating reuse) (a) within Stack Overflow posts, (b) between Stack Overflow and popular Java OSS projects, and (c) between the projects. Outcomes reveal that there was up to *3.3%* code reuse within Stack Overflow, while *1.0%* of Stack Overflow code was reused in recent popular Java projects and *2.3%* in those projects that were more established. Reuse across projects was much higher, accounting for as much as *77.2%*. Our outcomes have implication for strategies aimed at introducing strict quality assurance measures to ensure the appropriateness of code reuse, and licensing requirements awareness.**

Keywords—Code reuse, Stack Overflow, Java projects, OSS, Q&A, Quality

I. Introduction

Quality plays a fundamental role in software success [30]. Thus, quality standards have been developed to provide guidance for software developers, covering the requirements for producing high quality, defect-free software [30, 31]. Under the ISO-9126 quality model, for example, it is stated that the quality requirements for software should cover efficiency, functionality, reliability, usability, reusability, and maintainability [9]. Such standards have also been the subject of previous academic studies (e.g., Singh et al. [22]).

With quality as an underlying motivator for instilling good software development practices while creating software, developers should be particularly conscious when employing code reuse from external sources (e.g., from open source (OS) portals) [29], which may impact software efficiency, functionality, reliability, usability, and maintainability. While code reuse allows for previously tested and quality-assured code to be implemented in a system, reusing code from untrusted sources may lead to system harm [16]. The implications of code reuse could be particularly significant for software maintainability, as poor knowledge of reused code at the time of software development will likely create challenges for future corrective and perfective actions. As discussed in Roy et al. [40], understanding the levels of reuse and cloning could be valuable for developers in terms of assisting with issues related to plagiarism, software evolution, debugging, code compaction, and security. Furthermore, Kashima et al. [36] noted that there are several OSS licenses that require software outcomes derived from original solutions to be published under the same license. This demands that developers are aware of the legal implications of the licenses under which OSS and code posted on other portals (such as Stack Overflow) are published. Additionally, businesses also need to be aware of the reuse occurring within outsourced development [20], as under these conditions they may face future legal challenges.

Code reuse is formally defined as "*the use of existing software or software knowledge to construct new software*" [15]. It is prevalent in many software, including those produced by top-tier software development companies such as Google [34]. Beyond such industry leaders, code reuse has been found to be exceptionally common in Mobile Apps, with some of these products consisting entirely of reused elements [13]. This high level of reuse seen in the practice of developers stems from the benefits it provides in terms of easily adding and enhancing system features [25]. The accessibility of readily available solutions to coding problems is highly attractive to both novice and experienced programmers [25]. In fact, in a study by Sojer et al. [21], the responses from 869 developers confirmed that they consider ad hoc reuse of code from the internet to be important for their work. Similarly, Heinemann et al. [18] also found that 90% of the OS projects they analyzed contained reused code, reiterating the point that code reuse is found extensively in many software systems.

The ease and attractiveness of code reuse has been particularly aided by readily accessible code fragments on Q&A websites, such as Stack Overflow[1]. Stack Overflow is a very popular Q&A website which allows members of the public to post development related questions and/or answers, with the answers often containing code fragments.

[1] http://www.stackoverflow.com

2377-5408/18/$31.00 ©2018 IEEE
DOI 10.1109/ASWEC.2018.00027

Recent evidence shows that the majority of the questions that are asked on Stack Overflow usually receive one or more answers [6], and this forum is often a substitute for official programming languages' tutorials and guides [24].

With both implications for software maintainability and licensing when reusing Stack Overflow code fragments, of interest to us is the potential effects reusing code from this portal could have on effort for future changes and correct use of license to avoid future legal issues. The aim of this paper is thus to investigate the levels of code reuse within Stack Overflow, and between Stack Overflow and OSS projects. We focus on the Java programming language, given its popularity[2], and the need to understand reuse beyond Python (Yang et al. [8]). With a strong body of knowledge around the scale of developers' reuse practices, team leaders may begin to introduce stricter quality assurance measures to ensure the appropriateness of reused code fragments. We thus answer five research questions in our portfolio of work. Firstly, we explore, *what is the extent of Java code reuse within Stack Overflow?* to understand how the community operates as an ecosystem in the provision of self-support (RQ1). Related to this question, we next explore, *what is the extent of code reuse between answers published under the same question in Stack Overflow?* to understand the degree of innovation (or lack thereof) that is prevalent on this platform (RQ2). Answers to these two questions are particularly useful for the software engineering community as within-source code migration is likely to increase the risk of incorrect author attribution, due to (a) having more copies in existence, and (b) increasing the number of 'steps' a piece of code could have taken from its origin to where it was found. This could in turn lead to unsuspecting license violations for those implementing these code snippets in OSS.

Our third research question, *what is the extent of code reuse between Stack Overflow and the current most popular Open Source Java Projects?* helps us to understand recent code reuse trends (RQ3). Related to this research question we examine, *what is the extent of code reuse between Stack Overflow and the all-time most popular Open Source Java projects?* to understand software practitioners' behavior to code reuse over time (RQ4). Additionally, we answer, *are there differences in the nature of reuse found between the different contexts in terms of scale and size?* to provide deeper evidence for the nature and ranges of code reuse between Stack Overflow and OSS projects (RQ5). Beyond understanding the extent of code reuse (or clones) existing between OSS and Stack Overflow, it is important to understand how practitioners' attitude towards this practice has changed over time. Our investigation led by the latter three questions will provide initial evidence on the extent of code reuse between projects developed more recently and those having existed for longer.

The remaining sections of this paper are organized as follows. We provide our study background in Section 2. We next provide our research setting in Section 3, before providing our results in Section 4. We then discuss our findings and their implications in Section 5, prior to considering threats to the study in Section 6. Finally, we provide concluding remarks and point to future research in Section 7.

[2] http://redmonk.com/sogrady/2016/02/19/language-rankings-1-16/

II. Background

Software practitioners would benefit from developing maintainable software systems that are free of code license violations, and thus, code reuse should be given serious consideration during development. Both of these topics (i.e., software maintenance and license) have been investigated to various extents, and their importance has been widely noted in the literature. Firstly, the maintainability of a software system is highly significant to all its stakeholders, especially when considering project lead-times and costs [9]. Maintainability refers to the likelihood of performing software improvements in a given period, and is said to become more difficult with the prevalence of code reuse [32]. Kamiya et al. [32] established that code reuse could introduce multiple points of failure if code fragments are 'buggy', and in fact, it has been noted that approximately half of the changes made to code clone groups are inconsistent [15].

The issue of code reuse and maintainability becomes more complex when the reused code is sourced from external sources (e.g., Stack Overflow). This is due to potential code incompatibility issues and sub-optimal solutions, which are often tied to a lack of developer understanding. Also, code fragments provided on Stack Overflow are largely written for accompanying some textual explanation, and not for immediate use as such. In fact, for many developers, online sources such as Stack Overflow are of utility, when they are faced with issues that require knowledge they do not possess. This brings into question their likely understanding of such code, which in turn brings into question the software's quality. Furthermore, security complications may arise, as evidence has shown that Stack Overflow portal includes insecure code [10].

An example of how catastrophic code reuse could be is illustrated by Bi [11]. This author shows that a piece of Stack Overflow code was used in the NissanConnect EV mobile app, which accidentally displayed a piece of text reading "App explanation: the spirit of stack overflow is coders helping coders". This example illustrates that code reused from Stack Overflow and other similar portals are not always examined thoroughly. Although this example illustrates a non-threatening issue, many similar cases could introduce security and functionality-related problems if not inspected properly. Thus, it is important to investigate and understand the extent of code reuse occurring between software systems and online code resources such as Stack Overflow.

Recently, several research studies have been conducted on the topic of code reuse and Stack Overflow. For instance, Abdalkareem et al. [25] investigated code reused from Stack Overflow in Mobile Apps and found that 1.3% of the Apps they sampled were constructed from Stack Overflow posts. They also discovered that mid-aged and older Apps contained Stack Overflow code introduced later in their lifetime. An et al. [19] also investigated Android Apps and found that 62 out of 399 (15.5%) Apps contained exact code clones; and of the 62 Apps, 60 had potential license violations. In terms of Stack Overflow, they discovered that 1,226 posts contained code found in 68 Apps. Furthermore,

126 snippets were involved in code migration, where 12 cases of migration involved Apps published under different licenses. Yang et al. [8] noted that, in terms of Python projects, over 1% of code blocks in their token form exist in both GitHub and Stack Overflow. At an 80% similarity threshold, over 1.1% of code blocks in GitHub were similar to those in Stack Overflow, and 2% of Stack Overflow code blocks were similar to those in GitHub.

In terms of attribution, in ensuring conformance to license requirements, Baltes et al. [27] found that 7.3% of popular repositories on GitHub contained a reference to Stack Overflow. In the context of Java projects, a minimum of two thirds containing copied code did not contain a reference to Stack Overflow. Additionally, only 32% of surveyed developers were aware of the attribution requirements of Stack Overflow. This could result in complicated legal issues for developers. In fact, the study of licensing violations is also the subject of previous research [4, 23, 21]. It has been noted that license violations occur frequently in OS projects [4], as well as in Q&A websites such as Stack Overflow, where the community itself has inquired about the issue [3]. As stated by German et al. [7], it is illegal for code fragments from one system to be implemented in another if their licenses are incompatible. As such, developers are required to be cautious with their work and should be aware of the legal consequences involved with code reuse from internet sources. Although license violations do not have direct implications for quality, it does pose potential legal problems, which could result in the removal of software and court costs. Additionally, from a software development perspective, licensing issues could result in further costs to resolve complications, implement system changes, and fix reputation damage.

Stack Overflow is covered under the CC BY-SA 3.0: Creative Commons Attribution-ShareAlike 3.0 license [2], and as such, developers have the right to transform and build upon the content on Stack Overflow. However, new software using Stack Overflow code must be distributed under the same license as the original. Furthermore, credit must be given to the specific answer on Stack Overflow, a link must be provided for the license, and the developer should specify if they introduced changes. Noticeably, code reuse from Stack Overflow has been shown to exist in various OSS projects, with varying amounts of reuse levels. The reused code, however, is not often acknowledged, and the lack of attribution results in license violations in many projects [3, 25]. As such, additional research is required to both validate and extend the current literature. We pursue this line of work in this study, in answering our five research questions (RQ1-RQ5 stated earlier).

III. Research Setting

A. Data Collection and Processing

To address the research questions posed three sets of data were extracted, including Stack Overflow code snippets and two sets of OSS projects' source code. For the purpose of this study each dataset was required to only contain Java files. To collect the necessary data, we utilized the Stack Overflow data dump, SourceForge, and GitHub. A key motivator for selecting these sources was their popularity in the programming community and their open access to data. The projects selected from SourceForge and GitHub were all based on popularity (both weekly and all time), resulting in projects being selected which were widely used and contributed towards. As such, we believe that the effects of code reuse would be more significant for these projects than for less popular ones.

Stack Overflow Java Snippets: The Java 'snippets' from Stack Overflow were extracted using the data explorer function to create the first dataset. Answer posts were then selected based on having at least one "<code>" tag and were filtered on the language Java. Of these answers, only those which were selected as accepted answers were kept, on the premise that such snippets will be trusted and thus reused. As a final filter, only answers from 2014 to 2017 were selected to ensure relevancy. This resulted in 117,526 answers. These answers were then separated into individual code snippets, based on each being within "<code>…</code>" tags. This resulted in 404,799 individual code snippets. Of these snippets, only those with more than one line of code were selected. Ultimately, 151,954 code snippets were extracted and saved as Java files, and 151,946 were analyzed since eight returned errors when they were processed.

Top Weekly OSS Projects: The second dataset of files extracted were projects with the greatest weekly popularity, with the specific week of sourcing starting on December 18, 2017. We extracted the top 10 weekly Java projects on SourceForge and GitHub. This resulted in a preliminary sample of 20 projects, in line with previous research done by Heinemann et al. [18] on Open Source Java projects. Each of these projects was investigated, and those containing at least one Java file was selected. Ultimately, 12 suitable projects were finally selected for the analysis, which contained a total of 16,617 Java files. Five files returned errors during processing, as reported in Table III.

All Time Most Popular OSS Projects: The final dataset covered projects that had the highest all-time popularity on SourceForge. As above, the top 20 projects were selected, and 16 were appropriate for the analysis (i.e., contained at least one Java source file). We did not extract projects from GitHub in this round given the richness of the projects that were extracted from SourceForge. The projects were filtered on popularity, as well as containing Java code. However, four projects were included in the subset which did not contain Java files, leaving 39,616 files. The final list of projects and their summaries can be found in Table IV with 39,558 Java files being used in our analyses after processing.

B. Tools and Techniques

To answer our research questions an appropriate clone (reuse) detection tool was required. We conducted a review of several tools, including NiCad [14], SourcererCC [12] and CCFinderX [32]. We selected CCFinderX given its performance and popularity among researchers [5, 25, 32]. Its token-based techniques for clone detection is computationally more efficient than alternative methods, it has a high recall rate, and is able to detect all hidden clones [5]. As discussed by Kamiya et al. [32], the software works by employing a lexical analyzer to create token sequences, after which it applies rule-based transformations to these sequences (based on the specific programming language).

The lexical analyzer is used to transform sequences of characters into sequences of tokens, which are word-like entities [33]. These entities can be identifiers, keywords, numbers, literals, operators, separators, or comments [33, 1]. The matching of clones is then computed using a suffix-tree algorithm, "in which the clone information is represented as a tree with sharing nodes for leading identical subsequences and the clone detection is performed by searching the leading nodes on the tree" [33].

When utilizing CCFinderX for the analyses, several parameters were configured. We followed previous recommendations and used CCFinderX default settings [25]. The minimum clone length, representing the absolute count of tokens, was set at its default value of 50. As such, code blocks were only considered if they contain at least 50 tokens. Additionally, the minimum unique token set value was configured as default (being 12). Hence, code blocks were only considered if it contains at least 12 unique tokens in addition to having an absolute minimum count of 50 tokens. The shaper level was also set at its default of 2. The shaper restricts code blocks from being considered a candidate clone if an outer block '}' splits the token sequence. The final two parameters were the 'P-match application' and the 'Pre-screening application'. The P-match application parameter is by default ticked, and denotes that variables and function names are not replaced with special characters. The Pre-screening application was, by default, not ticked, as we wanted to retain all clone instances. Pre-screen is ticked to filter outcomes where there are visually too many code clones. The output from CCFinderX includes both file metrics and clone metrics. The file metrics provide file-level insights into the data, whereas the clone metrics provide information regarding clone sets. One set exists for each unique group of clones. As such, a clone set will contain a minimum of 2 code blocks. Additionally, we were able to identify the number of files containing clones and clone-sets present in different files in the data (refer to Figure 1 for example).

In order to determine the extent of code reuse occurring within files, between files, and between projects/datasets, the Radius metric (RAD) of CCFinderX was utilized.

```
58  public int hashCode() {
59      final int prime = 31;
60      int result = 1;
61      result = prime * result + v;
62      return result;
63  }
64
65  @Override
66  public boolean equals(Object obj) {
67      if ( this == obj ) {
68          return true;
69      }
70      if ( obj == null ) {
71          return false;
72      }
73      if ( getClass() != obj.getClass() ) {
74          return false;
75      }
76      final TestInnerClass other = (TestInnerClass) obj;
77      if ( v != other.v ) {
78          return false;
79      }
80      return true;
```

```
144 @Override
145 public int hashCode() {
146     final int prime = 31;
147     int result = 1;
148     result = prime * result + color;
149     return result;
150 }
151 @Override
152 public boolean equals(Object obj) {
153     if (this == obj)
154         return true;
155     if (obj == null)
156         return false;
157     if (getClass() != obj.getClass())
158         return false;
159     PaletteColor other = (PaletteColor) obj;
160     if (color != other.color)
161         return false;
162     return true;
163 }
164
165 public List<Integer> indices = new ArrayList<>();
166 }
```

Figure 1. Example of two files with clone-sets

After performing the analysis, clone-sets were selected based on their specific RAD values, and in turn these were used to select the individual files involved. The RAD metric, as defined by Kamiya et al. [32], gives an indication of the maximum distance to a common directory between the files involved in a clone-set. As such, clones found within the same file will have a Radius of 0, and clones found between two files in the same directory will have a Radius of 1, and so on.

C. Measures for Answering RQs

To answer the first four research questions posed (RQ1-RQ4), five analyses were performed. These analyses involve calculating the following metrics: Firstly, the number of files containing at least one clone is computed. Secondly, by using the previous measure, we got a measure of the percentage of files containing clones. This allows us to compare our results with those from similar studies, such as that of Yang et al. [8]. Thirdly, by summing the population variable (pop) of each clone-sets we identified the total number of clones present in the files. Fourthly, the total number of clone-sets reveals all unique clones. Fifthly, among these clone sets we identified which clones involved more than one file.

To answer ***RQ1***. *What is the extent of Java code reuse within Stack Overflow?*, all Stack Overflow files were stored in the same directory when the CCFinderX was executed, and as such a Radius of 1 was used to identify between-file clone-sets. Answering the second research question (***RQ2***. *What is the extent of code reuse between answers published under the same question in Stack Overflow?*) required Stack Overflow files to be stored in separate directories based on the questions under which they were posted. As such, a Radius of 1 would indicate that clones exist between answers for the same question, and a Radius of 2 would indicate that clones exist between answers for separate questions. Having a Radius of 2, however, does not imply that intra-question clones (i.e., clones under the same question) do not exist, it simply implies that a clone is also found between questions. This can hide intra-question clones, and as such a manual inspection was performed on the clone-sets with a Radius of 2 to identify intra-question clones hidden by the maximum Radius value. Figures 2 and 3 demonstrate the situation, where both cases have a Radius of 2, however only one (Figure 2) has an intra-question clone. The code piece, denoted by 'A', is found under the same question (Question 1).

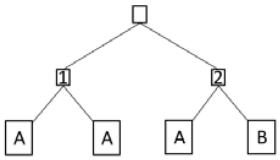

Figure 2. Clone-set with RAD of 2 containing intra-question clone

To answer the third (***RQ3***. *What is the extent of code reuse between Stack Overflow and the current most popular Open Source Java Projects?)* and fourth (***RQ4***. *What is the extent of code reuse between Stack Overflow and the all-time most popular Open Source Java projects?*) questions each project's files were extracted and saved under the same directory. Furthermore, the Stack Overflow files were saved two directories away, which allowed us to identify clone-sets with clones found between Stack Overflow and a project(s) using a Radius value of 2. The primary measurements required to answer the research questions includes the total number of files containing at least one clone, the total number of clones present in these files, and

the number of unique clones. ***RQ5***. *Are there differences in the nature of reuse found between the different contexts in terms of scale and size?* was answered through follow up statistical analyses involving the outcomes above.

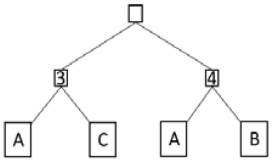

Figure 3. Clone-set with RAD 2 containing no intra-question clone

D. Reliability Checks

To ensure that the results obtained from our analyses were reliable, we conducted a manual investigation of 60 clone pairs detected by CCFinderX. Initially author AL (first author) performed the checks, which were then discussed with author SAL (second author) who triangulated the outcomes and provided confirmation. Within the sample of the 60 clone pairs, 20 were randomly obtained from the Stack Overflow analysis in Section IV (A), 20 from Section IV (C – a), and 20 from Section IV (D – a). For each selected clone-pair, it was determined to what extent the two pieces of code were similar, and the nature of the code was also recorded (i.e., is it a class, method, or piece of code within a method that was detected as being a clone). The extent to which clones were similar was rated either 'Exact', 'High', or 'Medium'. For those rated as 'Exact', the code in question would be identical copies, including all identifiers, the structure, and the functionality. For those rated as 'High', the primary difference between the two pieces of code would be the identifiers. Finally, those ranked as 'Medium' were considered to still be similar in structure, although identifiers, minor pieces of data structures, and minor pieces of functionality may be different. The results from the analyses are given in Tables I and II, where Table I reflects the number of clone pairs considered similar to a given extent, and Table II displays the nature of code elements detected in the sample.

TABLE I. MANUAL CHECK OF DETECTED CLONE SIMILARITY

Similarity	SO	SO & All Time Most Popular	SO & Current Most Popular	Total
Exact	10	6	3	19
High	10	12	13	35
Medium	0	2	4	6

TABLE II. CODE CLONES ELEMENTS

Nature of Code Element	SO	SO & All Time Most Popular	SO & Current Most Popular	Total
Class	5	0	1	6
Method	5	6	8	19
Part of Method	10	14	11	35

Our results show that it is highly plausible that these pieces of code could have been copied directly, or at least have been adapted to fit the software in question (refer to Table I for details). Furthermore, Table II shows that the majority of clones were code found within methods. Thus, it appears that if a developer is to copy a piece of code from Stack Overflow, then it is likely that this code would provide some additional functionality to a method.

IV. RESULTS

A. Java Code Reuse within Stack Overflow (RQ1)

Our analysis of the Stack Overflow files revealed that, overall, 5,041 files (out of 151,946) contained at least one clone (or were reused). Thus, *3.3%* of Stack Overflow Java code snippets have a duplicate found elsewhere in Stack Overflow. Furthermore, it was observed that within the 5,041 files, a total of 8,786 clones were present, indicating that some contained multiple clones. In terms of clone sets, 3,530 unique code snippets were observed to have clones. However, when focusing on clones found in at least two files, this number reduced to 2,338. As a result, we are able to determine that there were potentially 2,338 unique license violations existing within the Stack Overflow files extracted (refer to Section II for Stack Overflow licensing requirements), and that these cumulatively appear in 5,863 places. The additional 1,192 (i.e., 3530 minus 2338) unique clones were found within the same files, and as such, do not present potential license violations as they are contained within the same answers by the same author.

B. Java Code Reuse between Answers on Stack Overflow (RQ2)

To further investigate code reuse within Stack Overflow we also looked at the amount of reuse occurring within answers given to the same questions. Our analyses reveal that of the 151,946 Stack Overflow files 2,666 contained clones found under the same question. This equates to *1.8%* of the total files, and implies that this amount of snippets had at least one clone (code duplication) published under the same question. Within these 2,666 files, a total of 3,559 clones were found, again indicating that some answers contained multiple clones. Out of the 3,559 clones discovered, the number of unique clones were found to be 1,763. Additionally, from the 2,666 Stack Overflow files containing clones, we were able to identify that they were present in the answers in responses to 1,207 unique questions (out of 46,082 in total). Hence, *2.6%* of Java related questions on Stack Overflow can be expected to contain two or more answers with the same code.

C. Code Reuse between Stack Overflow and Current Popular Projects (RQ3)

a) Stack Overflow and Project Reuse Analysis: The analysis of the Stack Overflow and our top weekly OSS projects revealed that 12,763 files (out of 168,558; five project files were removed by CCFinderX's due to errors) contained at least one clone. Based on this result we observed that *7.6%* of the files under consideration contain at least one clone. Of the 12,763 files, a total of 5,447 of these were Stack Overflow files (out of 151,946 files), and 7,316 were top weekly OSS project files (out of 16,612 files). This indicates that when introducing the project files, 406 additional Stack Overflow files contain clones (refer to Section IV (A)). This implies that these 406 Stack Overflow files contain code that is not found anywhere else on Stack Overflow, with the clones being solely between Stack Overflow and at least one project. Additionally, the project files with clones account for *44%* of the total project files, which, as a proportion is much greater than that of the Stack Overflow files (just *3.3%*). This is primarily believed to be a

result of the size of the project files, with their average token size being 617, compared to a much smaller 48 for the Stack Overflow files. We performed further probing of the data, observing that in the 12,763 files containing at least one clone, 21,893 clone sets existed. In other words, there were 21,893 unique code snippets which have at least one clone. Of these, a smaller number of clone sets contained clones found in both Stack Overflow and top weekly OSS project files. This figure is 223, indicating that 223 unique code snippets are found between the Stack Overflow and project files. These clones cumulatively appear in 1,627 files (*1.0%* of 168,558), with each appearing in an average of 7.3 files. In total, these 223 unique code snippets appear 1,995 times.

b) Inter-Project Reuse Analysis: Of the 12,763 files containing clones, a total of 75,959 clones were discovered within these. When the project files were analyzed independently, it was found that 7,287 (i.e., *57%*) of the project files contained clones among themselves, giving an average of 2,979.1 clones per project (as depicted in Table III). Additionally, when investigating clone-sets, we observe 212 clones in at least two projects, with these appearing 1,995 times. Further probing also revealed that 29 project files (out of 7,316) contained clones that are only found in Stack Overflow files, and not in any other project. In other words, these 29 clones are found in a one to one fashion between one project and Stack Overflow, and as such they are most likely to have migrated directly between Stack Overflow and a project, since there is no evidence of these originating internal to the project. The direction of this migration, however, is not known, although independent of these situations, our reliability checks above show that there were no attributions, and thus, licensing issues could arise.

D. Code Reuse between Stack Overflow and All-Time Most Popular Projects (RQ4)

a) Stack Overflow and Project Reuse Analysis: The analysis of the Stack Overflow and all time most popular Java projects revealed that overall 24,537 files (out of 191,504; 58 project files were removed by CCFinderX's due to errors) contained at least one clone. Based on this result we observe that approximately *12.8%* of the files under question contain at least one clone. However, only 5,554 Stack Overflow files contained a clone, which is 513 more than when Stack Overflow files were considered on their own. On the other hand, 18,983 project files (out of 39,558 files) contained at least one clone, which is approximately *48%* of the total project files. Again, it should be noted, that the average length of a project file was 652 tokens. Furthermore, of the 24,537 files containing at least one clone, 51,282 clone sets existed. In other words, there were 51,282 unique code snippets which had at least one clone. Of these, a smaller number of clone sets contain clones found in both Stack Overflow and the projects. This figure is 450, indicating that 450 unique code snippets were found between the Stack Overflow and project files. These clones cumulatively appear 4,334 times (*2.3%* of 191,504), or in 6.4 files on average.

b) Inter-Project Reuse Analysis: Within the 24,537 files a total of 245,750 clones were discovered. Additionally, when analyzed independently, it was found that 18,935 of the project files contained clones among themselves (i.e., 77.2%), giving an average of 9,186.9 clones per project (as depicted in Table IV). Additionally, when investigating clone-sets, it was found that 726 clones were found it at least two projects, with these appearing 6,377 times. We noticed that 48 project files (out of 18,983) contained clones that are only found in Stack Overflow files, and not in any other project. As above, these 48 files are found directly between one project and Stack Overflow, and as such is highly likely to have migrated directly between Stack Overflow and a project.

TABLE III. SUMMARY OF THE TOP WEEKLY JAVA PROJECTS (INTER-PROJECT)

Project	Number of Java files	Average number of tokens/file	Number of clones	Number of files with clone/s
Awesome Java Leetcode	57	220	18	15
Dubbo	1327	498.4	1996	486
Elastic-Search	5576	936.7	13823	2869
Java Design Patterns	1018	120.5	320	189
Apache OpenOffice	3966	673	13674	2327
Proxyee down	25	342.1	3	3
Qmui Android	172	842.9	225	81
Sap NetWeaver Server Adapter for Eclipse	239	713.4	2906	140
Sofia	17	252.3	11	9
Spring Boot	3799	277.7	2356	1037
Timber	164	673.2	230	71
YiZhi	252	286.6	187	89
Total	**16612**	**5836.8**	**35749**	**7316**
Average/mean	**1384.3**	**486.4**	**2979.1**	**609.7**

E. Contextual Differences in Scale and Size of Reuse (RQ5)

In addition to the findings above, the results displayed in Table V and Figure 4 show that the sizes of clones found within the various contexts are different. Of primary interest is the larger mean sizes of the clones within Stack Overflow (refer to boxplots in Figure 4-A, B). These larger sizes suggest that there is a likelihood of the clones detected being true positives, i.e., they are indeed evidence of reuse where entire snippets are copied. Additionally, the median and upper quartile of the top weekly Java projects clone sizes are greater than that of the other four contexts where project files were included. This is displayed in Figure 4, graphs C, D, E, and F; where D can be seen to have a greater median and upper quartile value. This indicates that newer projects are constructed to a greater extent from reused elements.

In Table V the average and maximum sizes of clones found within the various contexts are presented. Interestingly, the clones in terms of their maximum sizes are smaller for the two analyses looking at Stack Overflow and OSS projects together (277 and 324 respectively). As such, we can see that the code clones found between Stack Overflow and OSS projects are at most 324 tokens in length. However, when looking at inter-project clones, we notice that the maximum values are much higher, with the biggest clone consisting of 1,369 tokens. This suggests that code reuse between projects involves copying of larger pieces of code, including entire components. In contrast to this, Stack Overflow code usually provides smaller code snippets as answers to specific coding questions, and so, evidence here may be linked to this reality.

To test for statistically significant differences between the six groups of measures (refer to Table V), in terms of clone sizes, a Kruskal-Wallis test was performed. This test was selected as it is non-parametric in nature (i.e., does not assume that the data follows a Normal distribution), and it does not require sample sizes to be equivalent [28].

TABLE IV. SUMMARY OF THE ALL-TIME MOST POPULAR JAVA PROJECTS (INTER-PROJECT)

Project	Number of Java files	Average number of tokens/file	Number of clones	Number of files with clone/s
Angry IP Scanner	219	397	102	48
Catacombae	91	758.6	223	33
Cyclops Group	2609	151.9	2545	1291
Eclipse Checkstyle Plug-in	1708	319	3115	782
Freemind	529	772	495	192
Hibernate	2392	285.6	2148	627
Hitachi Vantara - Pentaho	24494	673.2	112415	12008
Libjpeg-turbo	12	2061.3	44	7
OpenCV	148	1003.9	508	94
Sap NetWeaver Server Adapter for Eclipse	239	713.4	2921	144
Sweet Home 3D	233	2408.3	1476	142
TurboVNC	245	886.5	495	114
Vuze – Azureus	3639	750	5784	1461
Webmin	42	1505.1	66	21
Weka	2803	1100	14185	1948
Xtreme Download Manager	155	806.4	468	71
Total	**39558**	**14592.2**	**146990**	**18983**
Average/mean	**2472.4**	**912**	**9186.9**	**1186.4**

TABLE V: CLONE SIZE STATISTICS

Data Group	Median	Mean	Max	Mean Rank
A. Stack Overflow	66	85.7	938	14869.7
B. Stack Overflow Intra-Answers	69	87.2	938	15480.4
C. Stack Overflow and Top Weekly	57	67.9	277	11014.3
D. Top Weekly	60	84.3	774	13478.7
E. Stack Overflow and Top All Time	58	71.2	324	11646.4
F. Top All Time	58	69.2	1369	11392.1

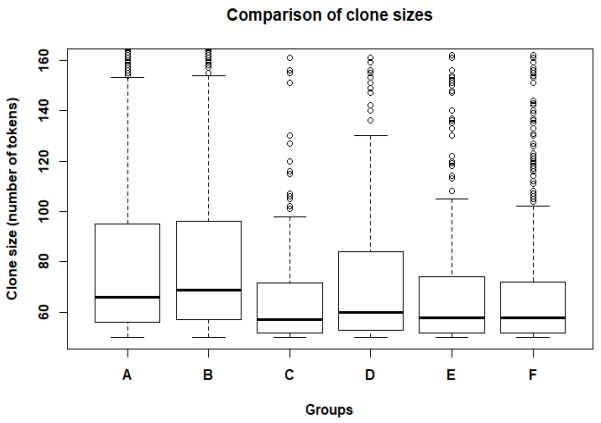

Figure 4. Clone size comparison between the groups defined in Table V

Our result reveals a statistically significant outcome (significance level = 0.05), providing evidence that our outcomes are different ($H(5) = 1409$, $p < 0.01$). Given this finding we further examined the distributions for A, B, D in Table V against others (C, E, F) with post hoc Kruskal-Wallis tests. Outcomes confirm that there were significantly bigger clones ($p < 0.05$) for Stack Overflow, Stack Overflow Intra-Answers and Top Weekly projects when compared to the other distributions. This, alongside the results in Table V and the boxplots in Figure 4, provide preliminary evidence that the nature of clones, in terms of their sizes, are different for different data sets. We thus plan further analyses to investigate why these differences exist.

V. DISCUSSION AND IMPLICATIONS

Discussion: Quality is an important element in all software development projects. In particular, the quality of freely available software should be a key consideration for its users. However, the migration of code between OSS projects and online Q&A platforms complicates such assessments. Stack Overflow as a platform, for instance, often acts as a medium through which code migrates between many projects, and as such, the quality of the code in many projects is influenced by factors that are beyond the control of their programmers. Furthermore, OSS projects are often published under specific licenses, which adds an additional level of complexity in terms of understanding their availability for reuse. In fact, users of the code published on Q&A platforms often lack the required understanding of the code, which can have direct implications for quality management if such code is reused in software projects. In order to investigate the extent of code reuse in these situations we focused on Java code from Stack Overflow and popular OSS projects. Here we revisit our outcomes to answer our five research questions (RQ1-RQ5).

RQ1. *What is the extent of Java code reuse within Stack Overflow?* Our results indicate that within Stack Overflow, approximately *3.3%* of the Java code sampled have at least one clone elsewhere on the website. Additionally, we found that up to 2,338 unique license violations could be present within these answers. This evidence duplicates that of Python code, which also revealed a 3.3% duplication [8]. It should be noted, however, that the Python code examined in Yang et al.'s [8] study was processed to remove the effects of white space and comments, which increase the performance of clone detection tools and lead to better comparisons. To this end, our outcomes is at best conservative, and so Java code reuse could be actually higher than 3.3% in Stack Overflow.

The results from our study, along with that of Yang et al. [8], indicate that code reuse is prevalent in Stack Overflow in both Java and Python contexts. The near identical results obtained by these two studies suggest that users and developers of the Stack Overflow platform should expect just over 3% of code on Stack Overflow to be duplicated. When considering the parameter settings for these code blocks to be considered candidate clones, it should be emphasized that these clones are of significant size (at least 50 tokens). Unlike many small snippets found on Stack Overflow, these clones meet the specified requirements set before the analysis, and as such, it is more likely that these code blocks are not clones by coincidence, rather they are reused. Hence, developers need to be cautious with reusing larger code blocks from Stack Overflow, and be prepared to rigorously evaluate such code before its usage. In addition,

instances of reuse demand proper attribution so that the community is aware of how Stack Overflow knowledge is recycled. We believe that a software tool could be of utility in terms of aiding developers wanting to evaluate the appropriateness of code for reuse, and also detecting exactly where such code originated from to help with correct attribution.

RQ2. *What is the extent of code reuse between answers published under the same question in Stack Overflow?* We observed that *1.8%* of all Java snippets (i.e., code in answers) have at least one clone within other answers provided for the same question. Our evidence also revealed that *2.6%* of questions sampled contain at least one clone pair between its answers. Furthermore, there were 1,763 potential unique license violations in our sample data. As with insights provided in response to RQ1, this outcome has implication for developers using Stack Overflow code in terms of the need to be aware of the rate of code duplication within Stack Overflow. With an overall duplication rate of 3.3%, we notice that a significant proportion of this duplication refers to clones between answers in different questions. As a result, developers may not give attribution to the original authors. Furthermore, in cases where these code blocks have migrated from external sources, having duplicates within Stack Overflow may make it more difficult to find these original sources. Without complete knowledge of the origin of reused code, developers may publish their OSS under different licenses, which will result in license violations. In fact, given the conservative settings used for our analyses, we anticipate that the reuse rate for smaller code snippets may be much higher. As such, if duplicated code can be identified by Stack Overflow, then the process of identifying the most appropriate solution (code) may be expedited, since users will be able to avoid duplicated answers. Having repeated duplicate answers may also result in convoluted pages, which could lead to slower problem solving for developers.

RQ3. *What is the extent of code reuse between Stack Overflow and the current most popular Open Source Java Projects?* Our evidence showed that between Stack Overflow and the top weekly Java projects, approximately 223 unique code snippets appeared in both sets of files. Between the Stack Overflow and project files, these snippets appeared in a total of 1,627 files. This evidence shows that, overall, *1.0%* of the project files contain one of these Stack Overflow clones. However, it should be noted that the percentage of project files containing clones is higher when compared to the percentage of Stack Overflow files that contained code. This outcome suggests that the current most popular open source Java projects tend to use code copied from Stack Overflow. In fact, within the projects, we discovered that approximately *57%* of these files contained a clone. These clones were found either within a single project, or between projects. In a study by Koschke et al. [26], they discovered that approximately 7.2% of all lines of code in Open Source Java projects were exact clones. These findings indicate that there is a high levels of code reuse and duplication within Open Source Java projects. Our findings suggest that an opportunity exists for developers to reduce their intra-project reuse, which could result in less maintainability issues. Furthermore, developers should also consider that code reuse is occurring between these projects, and as such, they should become acquainted with licensing requirements (refer to Section II).

RQ4. *What is the extent of code reuse between Stack Overflow and the all-time most popular Open Source Java projects?* When we compared Stack Overflow Java code against the all-time most popular OSS projects on SourceForge we observed that 450 unique code fragments were evident in both datasets, and that these appear in 4,334 files in total. This evidence shows that approximately *2.3%* of the files sampled contained at least one clone, and that there is one unique clone for every 54.5 project files. In fact, the proportion of project files containing clones was quite high, with approximately *77.2%* containing clones when excluding the Stack Overflow files.

Considering our outcomes against those of previous work [26], where 7.2% of code reuse was found, we believe that code reuse is high in popular Open Source Java projects. Interestingly, the percentage of files containing clones is higher for the all-time most popular projects, when compared to the newer, top weekly projects. It is thus more likely that code copied from these projects could have originally came from a different source, hence, creating a nested code reuse situation. Furthermore, the developers of these systems may potentially benefit from reducing the amount of reused code, thus improving the maintainability of their projects.

RQ5. *Are there differences in the nature of reuse found between the different contexts in terms of scale and size?* Our results show that there are differences in the sizes of clones found across our datasets. Our evidence shows that when reuse was done in Stack Overflow most of the snippets were copied. We also observed that current popular Java projects had a greater extent of reused elements from other projects. We believe that newer projects may be constructed more commonly from whole elements of other projects, i.e., the mean clone length is greater than that of the 'Top All-Time' group in Table V, possibly due to the availability of these elements, or perhaps developers are more willing to reuse in recent times. Similar outcomes were reported for Android Mobile Apps [25], which tends to dominate recent application development environments. Evidence here indicates that developers' behaviors are potentially changing, as we are seeing them incorporate larger pieces of copied code into their work. As such, the effects, both negative and positive, resulting from copying code will be amplified for these projects. In situations where the copied code is well explained in the respective sections on websites such as Stack Overflow, it could lead to better quality software, since the functionality is well understood, tested, and documented by developers. However, if larger pieces of code are copied and pasted without having sufficient accompanying documentation (e.g., comments), then it is likely that the software in question will contain code which is not understood by developers, thus bringing into questions its functionality, reliability, debuggability, and overall quality.

Our results also show a great degree of code duplication between all-time popular OSS projects, and, in fact, the scale and size of reuse was generally higher between OSS projects. This evidence is understandable given that Stack Overflow is generally known for shorter code snippets aimed at answering specific questions. Code duplication

between projects was possibly driven by the use of common third party libraries, but could also be through intentional duplication of similar functionalities. The fact that Stack Overflow snippets were also copied suggests that reuse may be a part of practitioners' culture. Thus, there are implications for making sure the correct license is used, and developers are aware of the strengths and weaknesses of the code that are copied. Furthermore, on the backdrop of the need for the community to develop high quality, maintainable and secured code, developers should carefully evaluate code that is reused.

Implications: Our investigation has shown that code clones do exist across Java-based projects and Stack Overflow. Having clones or duplicates within a system is unavoidable, since many software elements often rely on the same functionalities. However, in cases where many code clones exist, it is possible that developers may experience negative side-effects. Firstly, it is important to understand that high levels of code cloning can have negative effects on software quality, in terms of inconsistencies in code. Studies have found that around half of the software projects investigated had clones which contained inconsistencies, i.e., clones are changed inconsistently, with many of these being unintentional [15, 37]. Furthermore, these works also found that between 3-23% of code clones represented a fault. Thus, it is important for developers to be aware of the levels of code clones that exist within their software. To this end, we believe that tracking clones could improve the overall quality of software. This notion of tracking clones, and thus, being more aware of them, have been shown to improve software debugging [38, 39]. Another implication of our findings relates to probable licensing violations. Copying code from other projects or websites such as Stack Overflow without adhering to licensing requirements may result in complicated legal issues, and thus developers should take caution when doing so.

VI. Threats to Validity

Our analyses were conducted with CCFinderX, which uses a token-based approach to identify clones. This technique itself has some limitations, including a lower precision rate compared to some alternative techniques, primarily Abstract Syntax Tree (AST) techniques [5]. Additionally, CCFinderX had preset parameter settings for its analyses. These parameters were given specific values, which were used to filter all texts in order to identify candidate clones. As such, the detection of clones was based on code meeting the set requirements given by CCFinderX, possibly leading to some clones being missed by the software. This is particularly important when considering that we worked with Stack Overflow data, with which we had an average file token size of 48. Thus, we can assume that some smaller snippets from Stack Overflow reused in our Open Source projects were not detected, and thus, our results could be conservative.

In fact, our reliability checks show that many clones were of smaller sizes (refer to Table II). However, as code chunks get smaller, the ability to trace these back to their original source becomes challenging. Smaller code fragments may also be labelled as clones accidentally. That said, our contextual analyses performed for reliability evaluation ascertained that code was duplicated, and that there were no attributions. This evidence thus confirms the potential for future maintenance and quality issues, and possible licensing complications.

Additionally, we did not introduce a time element to determine the direction of reuse. As such, we cannot make conclusive statements regarding the temporal copying of code from Stack Overflow into OSS projects, and in terms of the direction of the copying (i.e., if code was copied from Stack Overflow to OSS projects, or OSS to Stack Overflow). Lastly, our sample of projects may not be representative of all software projects, and as such a large-scale study may produce more generalizable insights. The total number of projects on SourceForge containing Java code alone is over 40,000, and GitHub has over 3.5 million available Java-based repositories. Thus, a larger study may help validate the results obtained from this study. However, the initial study completed here reflects the findings from highly-used projects, making code reuse an important element to consider.

VII. Conclusion and Future Research

There is an imperative that the software engineering community develop and deliver high quality software. Improper code reuse as a practice may create barriers to the delivery of high-quality software however, and particularly in terms of software maintainability and confirming to legal requirements. With code reuse being a popular practice in the software engineering community, and Q&A forums such as Stack Overflow fueling this practice, it is pertinent to understand how this practice could affect future software maintenance and correct use of license to avoid legal issues. Towards this goal, we investigated the levels of code reuse within Stack Overflow, and between Stack Overflow and popular OSS projects.

Our findings have indicated that clones (reuse) do exist in all of the examined contexts (within Stack Overflow, between Stack Overflow and OSS, and between OSS), with numerous cases of code duplication detected in each setting. Outcomes in the work show that projects are all highly likely to contain code that has been copied from sources external to their own code. Additionally, our findings are similar to the research conducted on mobile apps and Python projects. As such, the levels of code reuse in these studies indicate that Java developers need to be made aware of licensing issues and the problems that could arise from ad-hoc copying. In particular, the quality assurance activities in software projects can be more comprehensive and could place greater emphasis on code reused from platforms such as Stack Overflow. This stands in agreement with [40], which discussed the benefits that code clone analysis can provide for software analysis. We also further believe that due to the increased amount of external code being integrated into projects, an even greater need exists for utilizing clone analysis software. If licensing knowledge and correct attribution is improved, then code fragments implemented from external sources will be less likely to cause licensing violations.

Our inter-project analyses showed that the top weekly Java projects had a greater average token size when compared to the all-time most popular Java projects. To further analyze this phenomenon, a time-based comparison of code reuse in OSS projects could be beneficial in

identifying the changes in reuse behavior over time. From our preliminary results it appears that newer projects have larger pieces of reused code, which could indicate that inter-project reuse of whole components is occurring. The work completed here can be replicated for a larger sample of projects in order to validate our results and assess the scale of reuse more generally. Additionally, research may look beyond the scope of OSS projects to contrast our findings with closed source projects. Our research may also be expanded to provide insights into the direction of migration of clones. An et al. [19] published results on code migration for Android mobile apps, and Inoue et al. [17] have developed a tool for tracking code in open source repositories, however dedicated work is required to investigate the direction of code migration from Stack Overflow (and other such portals) to OSS projects.

REFERENCES

[1] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: principles, techniques, and tools*. Harlow, Essex: Pearson, 2014.

[2] Anon. Creative Commons License Deed. Available: https://creativecommons.org/licenses/by-sa/3.0/, Feb. 2018.

[3] Anon. *Do I have to worry about copyright issues for code posted on Stack Overflow?* Available: http://meta.stackexchange.com/questions /12527/do-i-haveto-worry-about-copyright-issues-for-code-posted-on-stack-overflow, Feb. 2018.

[4] A. Mathur, H. Choudhary, P. Vashist, W. Thies, and S. Thilagam. "An Empirical Study of License Violations in Open Source Projects," *presented at the 35th Annual IEEE Software Engineering Workshop*. DOI:http://dx.doi.org/10.1109/sew.2012.24, 2012.

[5] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming,* vol. 74, pp. 470–495, 2009.

[6] J. Cordeiro, B. Antunes, and P. Gomes, "Context-based recommendation to support problem solving in sof. Development," *In Proceedings of 3rd Int.Workshop on RSSE*. 2012.

[7] D. M. German, M. Di Penta, Y-G. Gueheneuc, and G. Antoniol, "Code siblings: Technical and legal implications of copying code between applications," *In Proc. of 6th Working Conference on Mining Software Repositories*, 2009.

[8] D. Yang, P. Martins, V. Saini, and C. Lopes, "Stack Overflow in Github: Any Snippets There?" *In Proc. of 14th International Conference on Mining Software Repositories* (MSR), 2017. DOI:http://dx.doi.org/10.1109/msr.2017.13

[9] E. Johansson, A. Wesslen, L. Bratthall, and M. Host. "The importance of quality requirements in software platform development-a survey," In *Proc. of 34th Annual Hawaii International Conference on System Sciences,* 2001.

[10] Felix Fischer et al, "Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security," *IEEE Symposium on Security and Privacy (SP)*, 2017.

[11] F. Bi, "Nissan app developer busted for copying code from Stack Overflow," May. 2016. Available: https://www.theverge.com/tldr/ 2016/5/4/11593084/dont-get-busted-copying-code-from-stack-overflow

[12] H. Sajnani, V. Saini, J. Svajlenko, C. K. Roy, and C. V. Lopes, "SourcererCC," In *Proc. of 38th International Conference on Software Engineering* – ICDE 16, 2016.

[13] I. J. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, and A. Hassan, "A Large-Scale Empirical Study on Software Reuse in Mobile Apps," *IEEE Software*, vol. 31, no. 2, pp. 78–86, 2014. DOI:http://dx.doi.org/10.1109/ms.2013.142

[14] J. R. Cordy and C. K. Roy, "The NiCad Clone Detector," *Presented at the IEEE 19th International Conference on Program Comprehension*, 2011.

[15] J. Krinke, "A Study of Consistent and Inconsistent Changes to Code Clones," *Presented at the 14th Working Conference on Reverse Engineering* (WCRE), 2007.

[16] J. C. Knight and M. F. Dunn, "Software quality through domain-driven certification," *Ann. Softw. Eng*., vol. 5, pp. 293-315, 1998.

[17] K. Inoue, Y. Sasaki, P. Xia, and Y. Manabe, "Where does this code come from and where does it go? — Integrated code history tracker for open source systems," In *Proc. of 34th International Conference on Software Engineering*, 2012.

[18] L. Heinemann, F. Deissenboeck, M. Gleirscher, B. Hummel, and M. Irlbeck, "On the Extent and Nature of Software Reuse in Open Source Java Projects," *Lecture Notes in Computer Science Top Productivity through Software Reuse*, pp. 207–222, 2011.

[19] L. An, O. Mlouki, F. Khomh, and G. Antoniol, "Stack Overflow: A code laundering platform?" In *Proc. of IEEE 24th* SANER, 2017.

[20] M. Sojer and J. Henkel, "Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments," *Journal of the Association for Information Systems*, vol. 11, pp. 868–901, 2010.

[21] M. Sojer and J. Henkel, "License risks from ad hoc reuse of code from the internet," *Communications of the ACM*, vol. 54, pp. 74, 2011.

[22] M. Singh, A. Mittal, and S. Kumar, "Survey on Impact of Software Metrics on Software Quality," *International Journal of Advanced Computer Science and Applications,* vol. 3, 2012.

[23] O. Mlouki, F. Khomh, and G. Antoniol, "On the Detection of Licenses Violations in the Android Ecosystem," In *Proc. of IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering* (SANER), 2016.

[24] P., L., A. Bacchelli, and M. Lanza, "Leveraging crowd knowledge for software comprehension and development," CSMR, *IEEE Computer Society*, 2013, p. 57–66.

[25] R. Abdalkareem, E. Shihab, and J. Rilling, "On code reuse from StackOverflow: An exploratory study on Android apps," *Information and Software Technology*, vol. 88, pp. 148–158, 2017.

[26] R. Koschke and S. Bazrafshan, "Software-Clone Rates in Open-Source Programs Written in C or C++," In *Proc. of IEEE 23rd* SANER, 2016.

[27] S. Baltes, R. Kiefer, and S. Diehl, "Attribution Required: Stack Overflow Code Snippets in GitHub Projects," In *Proc. of IEEE/ACM 39th International Conference on Software Engineering Companion* (ICSE-C), 2017.

[28] S. Sawilowsky and G. Fahoome, *Kruskal-Wallis Test: Basic*, Wiley StatsRef: Statistics Reference Online, 2014.

[29] S. Haefliger, G. Von Krogh, and S. Spaeth, "Code Reuse in Open Source Software," *Management Science,* vol. 54, pp. 180-193, 2008.

[30] S. H. Kan, *Metrics and Models in Software Quality Engineering (2nd ed.),* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[31] V. Suma and T.R. Gopalakrishnan nair, "Effective Defect Prevention Approach in Software process for Achieving Better Quality levels," *World Academy of Science, Engineering and Technology,* vol. 42, pp. 258-262, 2008.

[32] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: a multilinguistic token-based code clone detection system for large scale source code," *IEEE TSE*, vol. 28, pp. 654–670, 2002.

[33] T. Ægidius Mogensen, "Lexical Analysis," *Introduction to Compiler Design Undergraduate Topics in Computer Science*, pp. 1–37, 2011.

[34] V. Bauer, J. Eckhardt, B. Hauptmann, and M. Klimek, "An exploratory study on reuse at google," In *Proc. of 1st International Workshop on Software Engineering Research and Industrial Practices* - SER&IPs, 2014.

[35] W.b. Frakes and K. Kang, "Software reuse research: status and future," *IEEE Transactions on Software Engineering,* vol. 31, pp. 529–536, 2005. DOI:http://dx.doi.org/10.1109/tse.2005.85

[36] Y. Kashima, Y. Hayase, N. Yoshida, Y. Manabe, and K. Inoue, "An Investigation into the Impact of Software Licenses on Copy-and-paste Reuse among OSS Projects," In *Proc. of 18th Working Conference on Reverse Engineering*, 2011.

[37] Elmar Juergens, Florian Deissenboeck, Benjamin Hummel, and Stefan Wagner, "Do code clones matter?" In *Proc. of IEEE 31st International Conference on Software Engineering,* 2009.

[38] Z. Li, S. Lu, S. Myagmar, and Y. Zhou, "CP-Miner: Finding copy-paste and related bugs in large-scale software code," *IEEE Trans. Softw. Eng*, vol. 32, pp. 176–192, 2006.

[39] L. Jiang, Z. Su, and E. Chiu, "Context-based detection of clone-related bugs," *In Proc. ESEC-FSE,* ACM, 2007.

[40] C. K. Roy, J. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming*, vol 74, no. 7, pp. 470-495, 2009.