Toxic Code Snippets on Stack Overflow

Chaiyong Ragkhitwetsagul, Jens Krinke, Matheus Paixao, Giuseppe Bianco, Rocco Oliveto

Abstract—Online code clones are code fragments that are copied from software projects or online sources to Stack Overflow as examples. Due to an absence of a checking mechanism after the code has been copied to Stack Overflow, they can become toxic code snippets, i.e. they suffer from being outdated or violating the original software license. We present a study incorporating state-of-the-art tools and techniques to automatically extract and filter online clone pairs between 72,365 Java code snippets on Stack Overflow and 111 open source projects in the curated Qualitas corpus. We analysed 2,302 non-trivial online clone candidates. Our investigation revealed strong evidence that 154 clones have been copied from a Qualitas project to Stack Overflow. We found 101 of them (66%) to be outdated and potentially harmful for reuse. Furthermore, we found 214 code snippets that could potentially violate the license of their original software. A survey of 201 high-reputation Stack Overflow answerers (33% response rate) showed that 131 participants (65%) have ever been notified of outdated code and 26 of them (20%) rarely or never fix the code. 138 answerers (69%) never check for licensing conflicts between their copied code snippets and Stack Overflow's CC BY-SA 3.0.

Index Terms—Code Clone Detection, Stack Overflow, Outdated Code, Software Licensing

1 Introduction

Stack Overflow is a popular online programming community with 7.6 million users, 14 million questions, and 23 million answers[1]. It allows programmers to ask questions and give answers to programming problems. The website has found to be useful for software development [15], [30], [42], [45], [46], [61], [62], [66] and also valuable for educational purposes [40]. On Stack Overflow, each conversation contains a question and a list of answers. The answers frequently contain at least one code snippet as a solution to the question asked. We found that the code snippets are usually not authored directly on the Stack Overflow website but copied from another location. A snippet in an answer could be copied and modified from a code snippet in the question, copied from the answerer's own code or from other locations including open source software (OSS) systems.

The process of posting and answering questions on Stack Overflow that involves the reuse (copying) of source code can be considered code cloning. Code cloning is an activity of reusing source code by copying and pasting. It normally occurs in software development and account from 7% to 23% of source code in typical software systems [7]. The benefits and drawbacks of clones are still controversial. Several authors state that clones lead to bug propagations and software maintenance issues [26], while some others suggest that clones are not harmful and can even be beneficial [27], [55].

Code cloning can also have side effects such as violating software licenses or introducing software vulnerabilities. Carelessly cloning code from one project to another project with a different license may cause a software license violation [19]. This also happens within the context of online Q&A websites such as Stack Overflow. An et al. [3] showed that 1,279 cloned snippets between Android apps and Stack Overflow potentially violate software licenses. Security is also among the main concerns when code is copied from an online source. For example, Stack Overflow helps developers to solve Android programming problems more quickly than other resources while, at the same time, offers less secure code than books or the official Android documentation [2].

We call code snippets that are copied from software systems to online Q&A websites (such as Stack Overflow) and vice versa as "online code clones". There are two directions in creating online code clones: (1) code is cloned from a software project to a Q&A website as an example; or (2) code is cloned from a Q&A website to a software project to obtain a functionality, perform a particular task, or fixing a bug. Similar to classic code clones, i.e. clones between software systems, online code clones can lead to license violations, bug propagation, introduction of vulnerabilities, and re-use of outdated code. Unfortunately, online clones are difficult to locate and fix since the search space in online code corpora is larger and no longer confined to a local repository.

To have a deeper insight on online code clones, we surveyed 201 high-reputation Stack Overflow answerers. The results of such a survey show that online code cloning occurs on Stack Overflow. Stack Overflow answerers frequently clone code from other locations, such as their personal projects, company projects, and open source projects, to Stack Overflow as a solution or a complement to a solution. The code cloning activity on Stack Overflow is obviously beneficial considered the popularity of Stack Overflow and its influence on software development [42], [45], [46]. On the other hand, there is also a downside caused by low quality, defective, and harmful code snippets that are reused without an awareness by millions of users.

One participant in our survey expresses his/her concerns about this:

> *"The real issue is less about the amount the code snippets on SO than it is about the staggeringly high number of software "professionals" that mindlessly use them without understanding what they're copying, and the only slightly less high number of would-be professionals that post snippets with built-in security is-*

1. Data as of 21 August 2017 from https://stackexchange.com/sites

```
/* Code in Stack Overflow post ID 22315734 */
public int compare(byte[] b1,int s1,int l1, ...) {
  try {
    buffer.reset(b1,s1,l1); /* parse key1 */
    key1.readFields(buffer);
    buffer.reset(b2,s2,l2); /* parse key2 */
    key2.readFields(buffer);
  } catch (IOException e) {
    throw new RuntimeException(e);
  }
  return compare(key1,key2); /* compare them */
}
```

```
/* WritableComparator.java (2014-11-21) */
public int compare(byte[] b1,int s1,int l1, ...) {
  try {
    buffer.reset(b1,s1,l1); /* parse key1 */
    key1.readFields(buffer);
    buffer.reset(b2,s2,l2); /* parse key2 */
    key2.readFields(buffer);
    buffer.reset(null,0,0); /* clean up reference */
  } catch (IOException e) {
    throw new RuntimeException(e);
  }
  return compare(key1, key2); /* compare them */
}
```

Figure 1: An example of the two code fragments of `WritableComparator.java`. The one from the Stack Overflow post 22315734 (left) is outdated when compared to its latest version in the Hadoop code base (right). Its Apache v.2.0 license is also missing.

> *sues. A related topic is beginners who post (at times dangerously) misleading tutorials online on topics they actually know very little about. Think PHP/MySQL tutorials written 10+ years after `mysql_*` functions were obsolete, or the recent regex tutorial that got posted the other day on HackerNew (https://news.ycombinator.com/item?id=14846506). They're also full of toxic code snippets."*

Using the same analogy, in this study, we are interested in mining Stack Overflow posts, detecting online code clones, and analysing the clones to reveal "toxic code snippets".

Toxic code snippets mean code snippets that are harmful for reuse and, in several cases, are caused by online code cloning. Stack Overflow code snippets cloned from open source software or online sources can become toxic when they are (1) outdated or (2) violating their original software license. Outdated code snippets can be harmful since they are not up-to-date with their originals and may contain defects. Code snippets from open source projects usually fall under a specific software license, e.g. GNU General Public License (GPL). If they are cloned to Stack Overflow answers without the license, and then flow to other projects with conflicting licenses, legal issues may occur.

We would like to motivate the readers by giving two examples of toxic code snippets. The first example is an outdated and potentially license-violating online code clone in an answer to a Stack Overflow question regarding how to implement `RawComparator` in Hadoop[2]. Figure 1 shows—on the left—a code snippet embedded as a part of the accepted answer. The snippet shows how Hadoop implements the `compare` method in its `WritableComparator` class. The code snippet on the right shows another version of the same method, but at this time extracted from the latest version (as of October 3, 2017) of Hadoop. We can see that they both are highly similar except a line containing `buffer.reset(null,0,0);` which was added on November 21, 2014. The added line is intended for cleaning up the reference in the `buffer` variable and avoid excess heap usage (issue no. HADOOP-11323[3]). While this change has already been introduced into the `compare` method several years ago, the code example in Stack Overflow post is still unchanged. In addition, the original code snippet of `WritableComparator` class in Hadoop is distributed with Apache license version 2.0 while its cloned instance on Stack Overflow contains only the `compare` method and ignores its license statement on top of the file. There are two potential issues for this. First, the code snippet may appear to be under Stack Overflow's CC BY-SA 3.0 instead of its original Apache license. Second, if the code snippet is copied and incorporated into another software project with a conflicting license, a legal issue may arise.

The second motivating example of outdated online code clones with more disrupting changes than the first one can be found in an answer to a Stack Overflow question regarding how to format files sizes in a human readable form. Figure 2 shows—on the left—a code snippet to perform the task from the `StringUtils` class in Hadoop. The code snippet on the right shows another version of the same method, but at this time extracted from the latest version of Hadoop. We can see that they are totally different. The `humanReadableInt` method is rewritten on February 5, 2013 to solve an issue of a race condition (issue no. HADOOP-9252[4]).

The two toxic code snippets in our examples have been posted on March 11, 2014 and April 9, 2009 respectively. They have already been viewed 259 and 2,886 times at the time of writing this paper (October 3, 2017). Our calculation finds that there will be a new viewer of the first toxic snippet approximately every 5 days compared to almost every day for the second one. A quick web search shows that the defective `humanReadableInt` method in the second example already appears in approximately 500 projects on GitHub. Considering the popularity of Stack Overflow, which has more than 50 million developers visiting each month[5], one toxic code snippet on Stack Overflow can spread and grow to hundred or thousand copies within only a year or two.

While research has mostly focused on reusing code snippets *from* Stack Overflow (e.g. [3], [30], [72]), fewer studies have been conducted on finding the origins of code examples copied *to* Stack Overflow and the awareness of Stack Overflow developers in doing so. Finding the origins of code examples reveals the problem of toxic code snippets caused by outdated code and software licensing violations.

2. http://stackoverflow.com/questions/22315734
3. https://issues.apache.org/jira/browse/HADOOP-11323
4. https://issues.apache.org/jira/browse/HADOOP-9252
5. Data as of 21 August 2017 from: https://stackoverflow.com

```
/* Code in Stack Overflow post ID 801987 */
public static String humanReadableInt(long number) {
  long absNumber = Math.abs(number);
  double result = number;
  String suffix = "";
  if (absNumber < 1024) {
  } else if (absNumber < 1024 * 1024) {
    result = number / 1024.0;
    suffix = "k";
  } else if (absNumber < 1024 * 1024 * 1024) {
    result = number / (1024.0 * 1024);
    suffix = "m";
  } else {
    result = number / (1024.0 * 1024 * 1024);
    suffix = "g";
  }
  return oneDecimal.format(result) + suffix;
}
```

```
/* StringUtils.java (2013-02-05) */
public static String humanReadableInt(long number) {
  return TraditionalBinaryPrefix.long2String(number,"",1);
}
```

Figure 2: An example of the two code fragments of `StringUtils.java`. The one from the Stack Overflow post 801987 (left) is outdated when compared to its latest version in the Hadoop code base (right). The toxic code snippet is outdated code and has race conditions.

It is probably more important than studying the effects of reusing Stack Overflow code snippets because it gives insights to the root cause of the problem and lays a foundation to an automatic technique to detect and report toxic code snippets on Stack Overflow to developers in the future.

This paper makes the following primary contributions:

1) **A manual study of online code clones:** We used two clone detection tools to discover 2,302 similar code snippet pairs between 72,365 Java code snippets obtained from Stack Overflow's accepted answers and 111 Java open source projects from the curated Qualitas corpus [65]. We manually classified all of them.
2) **An investigation of toxic code snippets on Stack Overflow:** Our study shows that from the 2,302 online clones, at least 330 have been copied from open source projects or external online sources to Stack Overflow, potentially violating software licenses. For 154 of them, we found evidence that they have been copied from a specific open source project. 101 of them we found to be outdated.
3) **Awareness of Stack Overflow answerers to toxic code snippets:** We performed an online survey and collected answers from 201 highly-ranked Stack Overflow users. We found that they cloned code snippets from open source projects to Stack Overflow answers. While Stack Overflow answerers are aware of their outdated code snippets, 19% of the participants rarely or never fix the code. 99% of the answerers never include a software license in their snippets and 69% never check for licensing conflicts between the cloned code snippets and Stack Overflow's CC BY-SA 3.0.
4) **An online code clone oracle:** The 2,302 manually investigated and validated online clone pairs are available for download[6] and can be used as a clone oracle.

6. https://ucl-crest.github.io/cloverflow-web

2 Empirical Study

We performed an empirical study of online code clones between Stack Overflow and 111 Java open source projects to answer the following research questions:

- **RQ1 (Online code clones):** *To what extent is source code cloned between Stack Overflow and open source projects?* We quantitatively measured the number of online code clones between Stack Overflow and open source projects to understand the scale of the problem.
- **RQ2 (Patterns of online code clones):** *How do online code clones occur?* We categorised online clones into seven categories allowing insights to how online code clones are created.
- **RQ3 (Outdated online code clones):** *Are online code clones up-to-date compared to their counterparts in the original projects?* We were interested in the outdated Stack Overflow code examples since users are potentially reusing them.
- **RQ4 (Software licensing violation):** *How often do license conflicts occur between Stack Overflow clones and their originals?* We investigated whether the reuse of online code clones can cause software developers to violate licenses.
- **RQ5 (Stack Overflow answerers' awareness):**
  1) How often are Stack Overflow answerers aware of the outdated code when they answer a question on Stack Overflow?
  2) How often are Stack Overflow answerers aware of licensing conflicts when they answer a question on Stack Overflow?
  We surveyed 607 high reputation Stack Overflow users to study their awareness of the two issues.

To answer these five research questions, we perform an empirical study to locate and study the online code clones between Stack Overflow and open source projects. We designed our study in 6 phases as depicted in Figure 3 where we build different data sets to answer each of our five research questions.

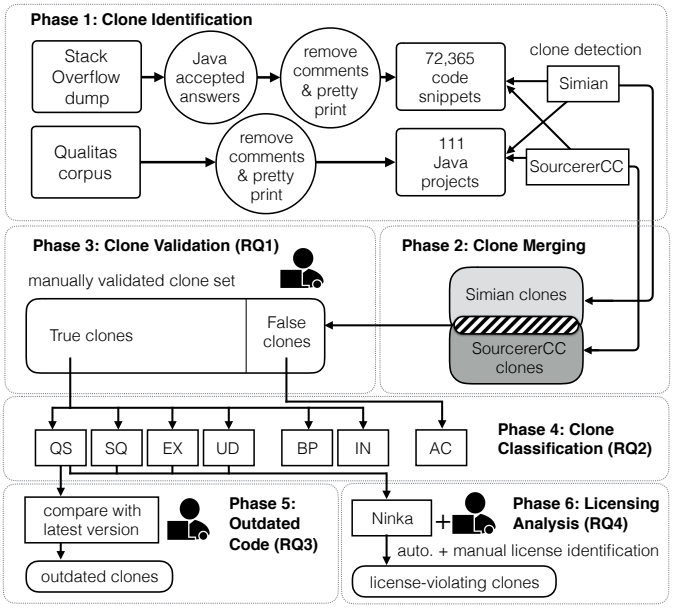


Figure 3: Experimental framework

2.1 Phase 1: Clone Identification

We rely on two source code data sets in this study: Java code snippets in answers on Stack Overflow and open source projects from the Qualitas corpus [65], as detailed next.

Stack Overflow: We extracted Java code snippets from a snapshot of a Stack Overflow dump[7] in January 2016. The data dump is in XML, and it contains information of posts (questions and answers). We were interested in code snippets embedded in posts which were located between `<code>...</code>` tags. A Stack Overflow thread contains a question and several answers. An answer can also be marked as an **accepted answer** by the questioner if the solution fixes his/her problem. We collected Java code snippets using two criteria. First, we only focused on code snippets in accepted answers. We chose the snippets in accepted answers because they actually solved the problems in the questions. Moreover, they are usually displayed just below the questions which makes them more likely to be reused than other answers. Second, we were only interested in code snippets of at least ten lines. Although the minimum clone size of six lines are usual in clone detection [7], [33], [70], we empirically found that snippets of six lines contain a large number of boiler-plate code of getters/setters, `equal` or `hashCode` methods, which are not interesting for the study. Each snippet was extracted from the dump and saved to a file. Moreover, we filtered out irrelevant code snippets that were part of the accepted answers but were not written in Java by using regular expressions and manual checking. Finally we obtained 72,365 Java code snippets containing 1,840,581 lines[8] of Java source code. The median size of the snippets is 17 lines.

Open source systems: We selected the established **Qualitas** corpus [65]. It is a curated Java corpus that has been used in several software engineering studies [6], [41], [64], [67]. The projects in the corpus represent various domains of software systems ranging from programming languages to visualisation. We selected the release 20130901r of the Qualitas corpus containing 112 Java open source projects. This release contains projects with releases no later than 1st September 2013. We intentionally chose an old corpus from 2013 since we are interested in online code clones in the direction from open source projects to Stack Overflow. The 20130901r snapshot provides Java code that is more than 2 years older than the Stack Overflow snapshot, which is sufficiently long for a number of code snippets to be copied onto Stack Overflow and also to observe if clones become outdated. Out of 112 Qualitas projects, there is one project, jre, that does not contain Java source code due to its licensing limitation [65] and is removed from the study. This resulted in 111 projects analysed in the study, for a total of 166,709 Java files containing 19,614,083 lines of code (see Table 1). The median project size is 60,667 lines of code.

Table 1: Stack Overflow and Qualitas datasets

Data set	No. of files	SLOC	Median
Stack Overflow	72,365	1,840,581	17
Qualitas	166,709	19,614,083	60,667

Clone Detection Tools: We use clone detection to discover online code clones. There are a number of restrictions in terms of choosing the clone detection tools for this study. The main restriction is due to the nature of code snippets posted on Stack Overflow, as most of them are incomplete Java classes or methods. Hence, a detector must be flexible enough to process code snippets that are not compilable or not complete blocks. Moreover, since the amount of code that has to be processed is in a scale of millions line of code (as shown in Table 1), a clone detector must be scalable enough to report clones in a reasonable amount of time. We have tried 7 state-of-the-art clone detectors including Simian [58], SourcererCC [56], NiCad [12], [54], CCFinder [26], iClones [22], DECKARD [25], and PMD-CPD [44] against the Stack Overflow and Qualitas datasets. NiCad failed to parse 44,960 Stack Overflow snippets while PMD CPD failed to complete the execution due to lexical errors. iClones could complete its execution but skipped 367 snippets due to malformed blocks in Stack Overflow data sets. CCFinder reported 8 errors while processing the two data sets. Although Simian, SourcererCC, and DECKARD could successfully report clones, we decided to choose only Simian and SourcererCC due to their fast detection speed. Moreover, Simian and SourcererCC complement each other as SourcererCC's clone fragments are always confined to method boundaries while Simian's fragments are not.

Simian is a text-based clone detector that locates clones at line-level granularity and has been used extensively in several clone studies [10], [36], [38], [50], [51], [70]. Furthermore, it offers normalisation of variable names and literals (strings and numbers) which enables Simian to detect literal clones (type-1) and parameterised clones (type-2) [7]. **SourcererCC** is a token-based clone detector which detects clones at either function- or block-level granularity. It can detect clones of type-1, -2 up to type-3 (clones with added and removed statements) and offer scalability against large code corpus [55], [56], [73].

7. https://archive.org/details/stackexchange
8. Measured by cloc: https://github.com/AlDanial/cloc

Table 2: Configurations of Simian and SourcererCC

Tool	Configurations
Simian (*S*)	Threshold=10, ignoreStringCase, ignoreCharacterCase, ignoreModifiers
SourcererCC (*SCC*)	Functions, Minimum clone size=10, Similarity=80%

Table 3: Number of online clones reported by Simian and SourcererCC

Tool	Total clone pairs	Average clone size
Simian	721	16.61
SourcererCC	1,678	17.86

We prepared the Java code in both datasets by removing comments and pretty-printing to increase the clone detection accuracy. Then, we deployed the two detectors to locate clones between the two datasets. For each Qualitas project, we ran the tools on the project's code and the entire Stack Overflow data. Due to incomplete code blocks and functions normally found in Stack Overflow snippets, the built-in SourcererCC's Java tokeniser could not parse 45,903 snippets, more than half of them. Nevertheless, the tool provides an option to plug in a customised tokeniser, so we developed a special Java tokeniser with assistance from the tool's creators. The customised tokeniser successfully processed all Stack Overflow snippets.

Simian did not provide an option to detect cross-project clones. Hence the Simian clone report was filtered to contain only clone pairs between Stack Overflow and Qualitas projects, removing all clone pairs within either Stack Overflow or within Qualitas. SourcererCC can detect cross-project clones between two systems so we did not filter the clones.

Clone Detection Configuration: We are aware of effects of configurations to clone detection results and the importance of searching for optimised configurations in empirical clone studies [50], [51], [52], [63], [69]. However, considering the massive size of the two datasets and the search space of at least 15 Simian and 3 SourcererCC parameters, we are unable to search for the best configurations of the tools. Thus, we decided to configure Simian and SourcererCC based on their established default configurations chosen by the tools' creators as depicted in Table 2. The two clone detectors complement each other by having Simian detecting literal copies of code snippets (type-1) and SourcererCC detecting clones with renaming and added/deleted statements (type-2, type-3). We made sure that they detected only non-trivial clones by increasing the minimum clone size to ten lines.

The number of online clone pairs reported are presented in Table 3. Simian reports 721 clone pairs while SourcererCC reports 1,678 clone pairs. SourcererCC reports more clones due to its capability of detecting clones of type-1 to type-3 while Simian only detects type-1 clones. The average clone size reported by Simian is 16.61 lines which is slightly smaller than SourcererCC (17.86 lines).

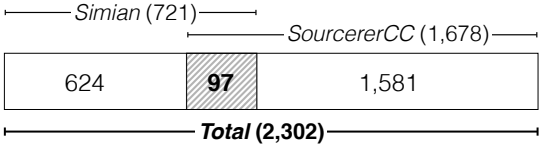


Figure 4: The result from clone merging using Bellon's ok-match criterion

2.2 Phase 2: Clone Merging

Clones from the two detectors can be duplicated. To avoid double-counting of the same clone pair, we adopted the idea of **clone agreement** which has been used in clone research studies [18], [52], [70] to merge clones from two data sets. Clone pairs agreed by both clone detection tools have a high likelihood to be duplicate and must be merged. To find agreement between two clone pairs reported by two different tools, we used the clone pair matching metric proposed by Bellon et al. [7]. Two clone pairs that have a large enough number of overlapping lines can be categorised as either a good-match or an ok-match pair with a confidence value between 0 and 1. Although good-match has stronger agreement than ok-match, we choose the ok-match criterion as our clone merging method because it depends on clone containment and does not take clone size into account. Clone containment suits our online code clones from two tools, Simian (line-level) and SourcererCC (method-level), better because Simian's clone fragments can be smaller or bigger than a method while SourcererCC's clone fragments are always confined to a method boundary.

We follow Bellon's original definitions of ok-match [7], which are based on how much two clone fragments *CF* are contained in each other:

$$contained(CF_1, CF_2) = \frac{|lines(CF_1) \cap lines(CF_2)|}{|lines(CF_1)|}$$

A clone pair *CP* is formed by two clone fragments CF_1 and CF_2, i.e. $CP = (CF_1, CF_2)$ and the *ok-value* of two clone pairs is defined as

$$\begin{aligned} ok(CP_1, CP_2) = min(&max(contained(CP_1.CF_1, CP_2.CF_1),\\ &contained(CP_2.CF_1, CP_1.CF_1)),\\ &max(contained(CP_1.CF_2, CP_2.CF_2),\\ &contained(CP_2.CF_2, CP_1.CF_2))) \end{aligned}$$

Two clone pairs CP_1 and CP_2 are called an *ok-match(t)* iff, for threshold $t \in [0, 1]$ holds

$$ok(CP_1, CP_2) \geq t$$

Using the ok-match criterion with a predefined threshold t of 0.7 similar to Bellon's study [7], we merge 721 clone pairs from Simian and 1,678 clone pairs from SourcererCC into a single set of 2,302 online clone pairs. There are only 97 common clone pairs between the two clone sets as depicted in Figure 4. The low number of common clone pairs is due to SourcererCC reporting clones with method boundaries while Simian is purely line-based.

Table 4: The seven patterns of online code cloning

Patt.	Description
QS	Cloned from Qualitas project to Stack Overflow (Q → S)
SQ	Cloned from Stack Overflow to Qualitas project (S → Q)
EX	Cloned from an external source to Stack Overflow (X → S)
UD	Cloned from each other or from an external source outside the project (unknown)
BP	Boiler-plate or IDE auto-generated
IN	Inheritance, interface implementation
AC	Accidental similarity, false clones

2.3 Phase 3-4: Validation and Classification

We used the 2,302 merged clone pairs for manual validation and classification. The validation and classification of the pairs was done at the same time. The clone validation process (phase 3 in Figure 3) involves checking if a clone pair is a true positive or a false positive. Moreover, we are also interested in the patterns of code cloning so we can gain more insights into how these clones are created (phase 4 in Figure 3).

Manual investigation: To mitigate the human error, we deployed two people in the manual clone investigation process. The first author, who is a research student working on clone detection research for three years, and the third author, who is a software engineering research student and familiar with code clones, took the role of the investigators performing a manual validation and classification of the merged clone pairs. The two investigators separately went through each clone pair candidate, looked at the clones, and decided if they are a true positive or a false positive and classified them into an appropriate pattern. After the validation, the results from the two investigators were compared. There were 338 (15%) conflicts between true and false clones (QS, SQ, EX, UD, BP, IN vs. AC). The investigators looked at each conflicting pair together and discussed until a consensus was made. Another 270 pairs (12%) were conflicts in the classification of the true clone pairs. Among these pairs, 145 conflicts were caused by one investigator being more careful than the other and being able to find evidence of copying while the other could not. Thus, resolving the conflicts lead to a better classification, i.e. from UD to QS or EX.

The online cloning classification patterns: We studied the eight patterns of cloning from Kapser et al. [27], [29] and performed a preliminary study to evaluate its applicability to our study. We tried to classify 697 online clone pairs from the reported clones in phase 1 using Kapser's cloning patterns. We found that Kapser's patterns are too broad for our study and a more suitable and fine-grained classification scheme is needed. After a preliminary study, we adopted one of Kapser's cloning patterns, *boiler-plate code*, and defined six new cloning patterns. The seven patterns include QS, SQ, EX, UD, BP, IN, and AC as presented in Table 4. Pattern QS (**Q**ualitas to **S**tack Overflow) represents clones that have evidence of being copied from a Qualitas project to Stack Overflow. The evidence of copying can be found in comments in the Qualitas source code or in the Stack Overflow post's contents. Pattern SQ (**S**tack Overflow to **Q**ualitas) is cloning, with evidence, in the opposite direction from Stack Overflow to a Qualitas project. Pattern EX (**E**xternal Sources) is cloning that has evidence of copying from a single or multiple external sources to Stack Overflow, and possibly also to a Qualitas project. Pattern UD (**U**nknown **D**irection) is cloning that creates identical or highly similar clones between Qualitas and Stack Overflow but where we could not find any attribution of copying. Pattern BP (**B**oiler-**P**late) represents clones containing boiler-plate. We define three cases of boiler-plate code and use in our classification as shown in Table 5. Our definition is specific to Java and more suitable to our study than the general definition in Kapser's [29]. Pattern IN (**In**heritance/Interface) is cloning by inheritance of the same super class or implementation of the same interface. These two activities usually result in similar overriding methods. The last pattern, AC (**A**ccidental **C**lones), represents accidentally similar clone pairs. These are mainly false positive clones from the clone detectors such as similar `try-catch` statements.

Table 5: The definition of boiler-plate code

Type	Description
API constraints	Similar code fragments are created because of a constraint by an API. For example, reading and writing to database using JDBC, reading and writing a file in Java.
Templating	An optimised or stable code fragment is reused multiple times. This also includes auto-generated code by IDE.
Design patterns	Java design patterns suggest a way of implementing similar pieces of code. For example, getters, setters, `equals`, `hashCode`, and `toString` method.

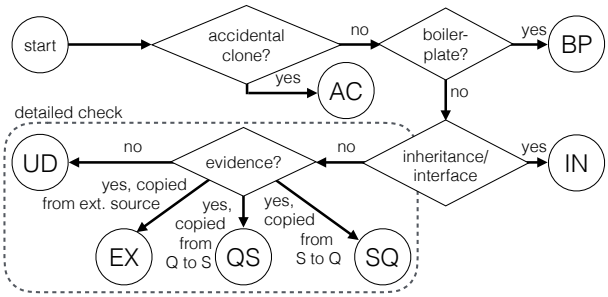


Figure 5: Online code clone classification process

The classification of the filtered online clone pairs followed the steps depicted in Figure 5. First, we look at a pair of clone fragments to see their similarity. If they are accidentally similar clones after code normalisation or false positive clones from the clone detection tools, we classify the pair into **AC**. If the two fragments are boiler-plate code, the pair is classified into **BP**. If they implement the same interface or inherited the same class and share similar overriding methods, the pair is classified into **IN**. If the pair is not **BP**, **IN**, or **AC**, we start a detailed investigation. We check the corresponding Stack Overflow post, read through it carefully and look for any evidence mentioning code copying. If evidence of copying has been found from a Qualitas project, the pair is classified in **QS**. In several occasions, we used extra information such as questions' contents, name of posters, and tags to gain a better understanding. On the other hand, if the source code from the Qualitas project

mentions copying from Stack Overflow, the pair is classified into **SQ**. If there is evidence of copying from an external source instead of a Qualitas projects, the pair is classified into **EX**. Lastly, if there is no evidence of copying in any direction but the clone fragments are highly similar, we classify them into **UD**.

2.4 Phase 5: Outdated Clones

Outdated code occurs when a piece of code has been copied from its origin to another location and later the original has been updated [71]. Usually code clone detection is used to locate clone instances and update them to match with the originals [7]. However, online code clones are more difficult to detect than in regular software projects due to its large search space and a mix of natural and programming languages combined in the same post.

To search for outdated online code clones, we focused on the **QS** clone pairs that were cloned from Qualitas to Stack Overflow and compared them with their latest versions. We downloaded the latest version of the Qualitas projects from their repositories on October 3, 2017. For each **QS** online clone pair, we used the clone from Qualitas as a proxy. We searched for its latest version by the file name and located the cloned region in the file based on the method name or specific code statements. We then compared the Stack Overflow snippet to its latest version line-by-line to find if any change has been made to the source code. We also made sure that the changes did not come from the modifications made to the Stack Overflow snippets by the posters but from the updates in the projects themselves. When we found inconsistent lines between the two versions, we used `git blame` to see who modified those lines of code and the timestamps. We also collected IDs of issue tracking information if the code change is linked to an automatic issue tracking system, such as Jira or BugZilla.

2.5 Phase 6: Licensing Analysis

Software licensing plays an important role in software development. Violation of software licenses impacts software delivery and also leads to legal issues [60]. One can run into a licensing issue if one integrates third-party source code into their software without checking. A study by An et al. [3] reports 1,279 cases of potential license violations between 399 Android apps and Stack Overflow code snippets.

We analysed licensing conflicts of the online clones in the **QS**, **EX**, and **UD** set. The licenses were extracted by *Ninka*, an automatic license identification tool [20]. Since Ninka works at file level, we report the findings based on Stack Overflow snippets and Qualitas source files instead of the clone pairs (duplicates were ignored). For the ones that could not be automatically identified by Ninka and have been reported as `SeeFile` or `Unknown`, we looked at them manually to see if any license can be found. For EX clone pairs that are cloned from external sources such as JDK or websites, we manually searched for the license of the original code.

2.6 Stack Overflow Developers' Survey

We support our findings of toxic code snippets on Stack Overflow by asking Stack Overflow users to take an online survey. The survey was used for assessing awareness of the developers on the two issues of outdated code and license-violating code snippets. We designed the survey using Google Forms by following the guidelines by Pfleeger and Kitchenham [31], [43]. The survey was completely anonymous and the participants could decide to leave at any time. It contained 11 questions. There were 7 Likert's scale questions, 3 yes/no questions, and one open-ended question for additional comments. The first two questions were mandatory while the other 9 questions were shown to the participants based on their previous answers. The full survey can be found in our research note [49].

We selected the participants for our survey based on their Stack Overflow reputation. On Stack Overflow, a user's reputation reflects how much the community trust them. A user earns reputations when he or she receives up votes for good questions and useful answers. Accepted answers receive more reputation score than questions, and regular answers[9]. Thus, Stack Overflow reputation is an indicator of user's skills and their involvement in asking and answering questions on the site. In this study, we call Stack Overflow users who have high reputation "Stack Overflow answerers."

The participants were invited to take the survey via email addresses available on their public Stack Overflow and GitHub profiles. We selected the answerers based on the all-time reputation ranking[10]. The invited participants had a reputation from 963,731 (the highest) to 6,999 and we sent out 607 emails (excluding undelivered ones, e.g. due to illegal email addresses). The survey was open for participation for two months, from 25 July to 25 September 2017, before we collected and analysed the responses.

3 Results and Discussion

We followed the 6 phases in the experimental framework (Figure 3) to answer our four research questions. To answer RQ1, we rely on the number of manually validated true positive online clone pairs in phase 3. We use the results of the manual classification by the seven patterns of online code cloning to answer RQ2 (phase 4). For RQ3, we looked at the true positive clone pairs that are classified as clones from Qualitas to Stack Overflow and checked if they have been changed after cloning (phase 5). Similarly, for RQ4, we looked at the license of each clone in the pattern QS, EX, UD and checked for a possibility of license violation (phase 6). We use the online survey to answer RQ5.

3.1 RQ1: Online Code Clones

To what extent is source code cloned between Stack Overflow and open source projects?

The statistics on clones obtained from the merged clone data set are presented in Table 6. Simian and SourcererCC reported clones in 460 snippets, approximately 0.6% of the 72,365 Stack Overflow snippets, associated with 59 Qualitas

9. Stack Overflow Reputation: https://stackoverflow.com/help/whats-reputation

10. Stack Overflow Users (data as of 25 July 2017): https://stackoverflow.com/users?tab=Reputation&filter=all

Table 6: Investigated online clone pairs and corresponding snippets and Qualitas projects

Set	Pairs	Snippets	Projects	Cloned ratio
Reported clones	2,302	460	59	53.10%
TP from manual validation	2,076	443	59	53.89%

projects. For the cloned Stack Overflow snippets, the average ratio of cloned code is 53.10% (i.e. the number of cloned lines of the cloned Stack Overflow snippet).

During the manual investigation of 2,302 clone pairs, we identified 226 pairs as being accidental clones (AC), i.e. false positives. After removing them, the set still contains 2,076 true positive clone pairs between 443 Stack Overflow snippets and 59 Qualitas projects. The average cloned ratio for the true positive clone pairs is 53.89%.

To answer RQ1, we found more than half (53%) of Qualitas projects to share similar code with Stack Overflow code snippets. In the manually confirmed data set of 2,302 clone pairs, we found 2,076 pairs as true positives which account for 59 Qualitas projects and 443 Stack Overflow code snippets.

3.2 RQ2: Patterns of Online Code Cloning

How do online code clones occur?

We performed a manual classification of the 2,302 merged clone pairs by following the classification process in Figure 5. The classification results are shown in Table 7 and explained in the following.

QS: Qualitas → Stack Overflow. We found 248 online clone pairs with evidence of cloning from Qualitas projects to Stack Overflow. However, we observed that, in some cases, a cloned code snippet on Stack Overflow matched with more than one code snippet in Qualitas projects because of code cloning inside Qualitas projects themselves. To avoid double counting of such online clones, we consolidated multiple clone pairs having the same Stack Overflow snippet, starting line, and ending line into a single clone pair. We finally obtained 154 QS pairs (Table 7) having unique Stack Overflow code snippets and associated with 23 Qualitas projects listed in Table 8. The most cloned project is hibernate with 24 clone pairs, followed by eclipse (21 pairs), jung2 (19 pairs), spring (17 pairs), and jfreechart (13 pairs). The clones are used as examples and are very similar to their original Qualitas code with limited modifications. Most of them have a statement in the Stack Overflow post saying that the code is "copied", "borrowed" or "modified" from a specific file or class in a Qualitas project. For example, according to the motivating example in Figure 1, we found evidence in the Stack Overflow Post 22315734 saying that *"Actually, you can learn how to compare in Hadoop from WritableComparator. Here is an examle that borrows some ideas from it."*

SQ: Stack Overflow → Qualitas. We found one pair with evidence of cloning from Stack Overflow post ID 698283 to `POIUtils.java` in jstock project. The user who asked the question on Stack Overflow is an author of jstock.

```
private Method findMethodToInvoke(Object test) {
  Method method = parameterTypeMap.get(test.getClass());
  if (method != null) {
    return method;
  }

  // Look for superclasses
  Class<?> x = test.getClass().getSuperclass();
  while (x != null && x != Object.class) {
    method = parameterTypeMap.get(x);
    if (method != null) {
      return method;
    }
    x = x.getSuperclass();
  }

  // Look for interfaces
  for (Class<?> i : test.getClass().getInterfaces()) {
    method = parameterTypeMap.get(i);
    if (method != null) {
      return method;
    }
  }
  return null;
}
```

Figure 6: The `findMethodToInvoke` that is found to be copied from Stack Overflow post 698283 to `POIUtils` class in jstock.

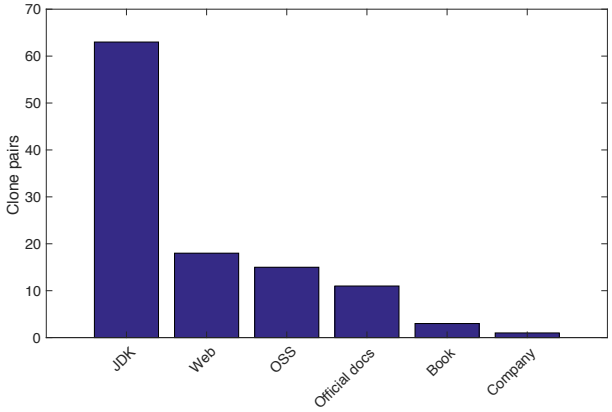


Figure 7: Original sources of EX clone pairs

The question is about determining the right method to call among 7 overloading methods of `setCellValue` during runtime. We could not find evidence of copying or attribution to Stack Overflow in jstock. However, considering that the 25 lines of code of `findMethodToInvoke` method depicted in Figure 6 in Stack Overflow is very similar to the code in jstock including comments, it is almost certain that the two code snippets form a clone pair. In addition, the Stack Overflow answer was posted on March 30, 2009, while the code in `POIUtils` class in jstock was committed to GitHub on the next day of March 31, 2009.

This very low number of SQ clone pair is very likely due to the age of the Qualitas corpus as another study [3] showed the presence of clones from Stack Overflow in newer open source data sets. This is expected and comes from our experimental design since we are more interested in cloning from Qualitas to Stack Overflow.

EX: External Sources. We found 199 clone pairs from external sources to Stack Overflow. After consolidating

Table 7: Classifications of online clone pairs.

Set	QS	SQ	EX	UD	BP	IN	AC	Total
Before consolidation	248	1	199	107	1,505	16	226	2,302
After consolidation	154	1	111	65	217	9	53	560

Table 8: Qualitas projects associated with QS and UD online clone pairs

QS		UD	
Project	Pairs	Project	Pairs
hibernate	24	netbeans	11
eclipse	21	eclipse	8
jung2	19	jstock	5
spring	17	compiere	5
jfreechart	13	ireport	4
hadoop	10	jmeter	4
log4j	8	jung2	3
tomcat	8	jhotdraw	3
struts	5	c-jdbc	3
lucene	4	log4j	3
weka	4	wct	2
junit	3	hibernate	2
poi	3	tomcat	2
compiere	2	spring	1
jasperreports	2	rssowl	1
jboss	2	mvnforum	1
jgraph	2	jfreechart	1
jstock	2	jboss	1
antlr	1	hadoop	1
itext	1	geotools	1
jgrapht	1	freemind	1
ant	1	findbugs	1
c-jdbc	1	cayenne	1

duplicated SO snippets due to multiple intra-clone instances in Qualitas, we obtained 111 EX pairs. We found evidence of copying from an external source to both Stack Overflow and Qualitas in 50 pairs. Each of the pairs contains statement(s) pointing to the original external location of the cloned code in Qualitas and Stack Overflow. In addition, we found evidence of copying from an external source to Stack Overflow, but not in Qualitas, in 61 pairs. Our analysis shows that the external sources fall into six groups as displayed in Figure 7. There are 63 EX online clone pairs copied from source code of Java SDK (e.g. java.util, javax.swing, javax.servlet), 18 pairs from websites, 15 pairs from open source systems not in Qualitas (e.g. Mozilla Rhino), 11 pairs from Java official documentations from Sun Microsystems or Oracle, 3 pairs from books, and 1 pair from a company project. For example, Stack Overflow Post 9549009 contains a code comment saying *"Copied shamelessly from org.bouncycastle.crypto.generators.PKCS5S2ParametersGenerator"* which is an open source project not included in the Qualitas corpus. Post 92962 includes a `VerticalLabelUI` class with a license statement showing that it is developed by a private company called Sapient. Post 12879764 has a text saying *"Code modified and cleaned from the original at Filthy Rich Clients."* which is a book for developing animated and graphical effects for desktop Java applications. Another example is a copy of code from a website in post 15260207. The text surrounding source code reads "*I basically stole this from the web and modified it slightly... You can see the original post here (http://www.java2s.com/Code/Java/Swing-JFC/DragListDemo.htm).*". Interestingly, the code is actually a copy from Sun Microsystems.

These findings complement a study of clones between software projects [63]. We found that cloning can also happen among different sources on the Internet just like software projects. The are 18 clone pairs that originated from programming websites including www.java2s.com and exampledepot.com. Moreover, there is one snippet which comes from a research website. We found that a snippet to generate graphical *Perlin noise* is copied from NYU Media Research Lab (http://mrl.nyu.edu/~perlin/noise/) website and is used in both Stack Overflow answer and the aoi project with attribution.

UD: Unknown Direction. We found 107 online clone pairs, reduced to 65 pairs after consolidating the clones, with no evidence of cloning between Qualitas and Stack Overflow but with a high code similarity that suggests cloning. The most cloned project is netbeans with 11 clone pairs. Most of the clones are a large chunk of code handling GUI components. Although these GUI clones might be auto-generated by IDEs, we did not find any evidence. The second most cloned project is eclipse (8 pairs), followed by jstock (5 pairs), a free stock market software, and compiere, a customer relationship management (CRM) system.

BP: Boiler-Plate. There were a large amount of boiler-plate clone pairs found in this study. We observed 1,505 such clone pairs and 217 after consolidation. The BP clone pairs account for 65% of all clone pairs we classified. The majority of them are `equals()` methods.

IN: Inheritance/interface. There were 16 clone pairs, 9 pairs after consolidation, found to be similar because they implement the same interface or inherit from the same class. An example is the two implementations of a custom data type which implements `UserType`. They share similar `@Override` methods of `deepCopy`, `isMutable`, `assemble`, `disassemble`, and `replace`.

AC: Accidental Clones. There were 226 accidental clone pairs, 53 after consolidation. Mainly, they are false positive clones caused by code normalisation and false type-3 clones from SourcererCC. Examples of the AC clone instances include `finally` or `try-catch` clauses that were accidentally the same due to their very small sizes, and similar `switch-case` statements.

To answer RQ2, we found 154 pairs with strong evidences to be cloned from 23 Qualitas projects to Stack Overflow, 1 pair was cloned from Stack Overflow to Qualitas, and 111 pairs were found to be cloned to Stack Overflow from external sources. However, the largest amount of the clone pairs between Stack Overflow and Qualitas projects are boiler-plate code (217), followed by 65 clone pairs with no evidence that the code has actually been copied, and 16 pairs of clones due to implementing the same interface

Table 9: Six code modification types found when comparing the outdated clone pairs to their latest versions

Modification	Occurrences
Statement modification	50
Statement addition	28
Statement removal	18
Method signature change	16
Method rewriting	15
File deletion	15

or inheriting the same class.

3.3 RQ3: Outdated Online Code Clones

Are online code clones up-to-date compared to their counterparts in the original projects?

We discovered 101 outdated online clone pairs out of 154 pairs. As shown in Figure 8, hibernate has the highest number of 19 outdated pairs, followed by 14 from spring, 13 from eclipse, and 9 from hadoop. Besides the two examples of outdated code in `WritableComparator` and `StringUtils` class from hadoop shown in Figure 1 and Figure 2, we also found a few outdated code elements which contained a large amount of modifications. For example, the code snippet in Stack Overflow post 23520731 is a copy of `SchemaUpdate.java` in hibernate. The code has been heavily modified on February 5, 2016.

We analysed code modifications which made Stack Overflow code outdated by going though commits and git blame information. The six code modification types found in the 101 outdated online clone pairs are summarised in Table 9. They include statement addition, statement modification, statement removal, method rewriting, API change (changing in method signature), and file deletion. We occasionally found multiple code modifications applied to one clone pair at the same time but at a different location. The most often code change found is statement modification (50 occurrences), followed by statement addition (28 occurrences), statement removal (18 occurrences), change of method signature, i.e. API change (16 occurrences), and method rewriting (15 occurrences). Moreover, in the 101 outdated pairs, we found 15 "dead" snippets. These snippets cannot be located in the latest version of the projects. For example, the snippet in Stack Overflow post 3758110, a copy of `DefaultAnnotationHandlerMapping.java` in spring, was deleted in the commit `02a4473c62d8240837bec297f0a1f3cb67ef8a7b` by Chris Beams on January 20, 2012, two years after it was posted.

Moreover, using the information in git commit messages, we can associate each change to its respective issues in an issue tracking system, such as Bugzilla or Jira. We found that in 58 cases, the cloned code snippets on Stack Overflow were changed because of a request in the issue tracking system. Since issue tracking systems are also used, besides bug reports, for feature request and feature enhancements, having an issue tracking ID can reflect that the change is important and not only a superficial fix such as code formatting.

Table 10 shows examples of the outdated online clones on Stack Overflow. The table displays information of the clones from both Stack Overflow and Qualitas side including the dates. We summarise the changes that make the clones outdated into three types, modified/added/deleted statements (*S*), file deletion (*D*), and method rewriting (*R*), along with the issue tracking number and the date of change. The complete set of 101 outdated online clones can be found on the study website.

The outdated online code clones cause problems ranging from uncompilable code (due to modifications and different API usage in the outdated code) to the introduction of vulnerabilities to the software [71]. An outdated code with a subtle change (e.g. Figure 1) may be copied and reused without awareness from developers. Moreover, an outdated code with a defect (e.g. a race condition problem in Figure 2) is clearly harmful to be reused. Although Stack Overflow has a voting mechanism that may mitigate this issue, the accepted answer is still used by naive developers who copy and reuse the outdated code.

For RQ3, our results show that 66% (101) of QS clone pairs on Stack Overflow are outdated. 86 pairs differ from their newest versions by modifications applied to variable names or method names, added or deleted statements, to a fully rewritten code with new method signatures. 15 pairs are dead snippets.

3.4 RQ4: Software Licensing Violations

Do licensing conflicts occur between Stack Overflow clones and their originals?

In our study, we reveal another type of toxic code snippets which is software licensing issues caused by code cloning to Stack Overflow. We found evidence that 154 pieces of code have been copied from Qualitas projects to Stack Overflow as examples. Another 111 pieces of code are cloned from external sources. Their status of accepted answers increase their chances of being reused. Even though most of the Qualitas projects came with a software license, we found that the license information were frequently missing after the code was copied to Stack Overflow. The licensing terms on top of source code files are not copied because usually only a small part of the file was cloned. In overall, we can see that most of the Stack Overflow snippets do not contain licensing terms while their clone counterparts in Qualitas projects and external sources do. Since licensing statement resides on top of a file, the results here are analysed at file level, not clone fragment, and clone pairs from the same file are merged. The summary of licensing information is listed in Table 11.

Compatible license: There are 41 pairs which have compatible licenses such as *Apache license v.2*; *Eclipse Public License v.1 (EPLv1)*; or a pair of *Creative Common Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)* vs. no license. These clones are safe for being reused. Since source code and text on Stack Overflow are protected by *CC BY-NC-SA 3.0* , we can treat the Stack Overflow code snippets without licensing information as having *CC BY-NC-SA 3.0* by default. The *CC BY-NC-SA 3.0* license is relaxed and it only requests an attribution when reused.

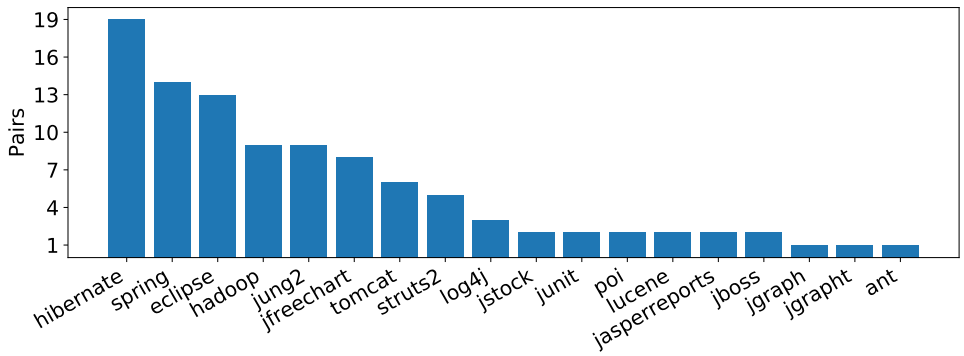


Figure 8: Outdated QS online clone pairs group by projects

Table 10: Examples of the outdated QS online clones

Stack Overflow		Qualitas						Changes		
Post	Date	Project	Ver.	File	Start	End	Date	Issue ID	Type*	Date
2513183	25-Mar-10	eclipse	4.3	GenerateToStringAction.java	113	166	5-Jun-13	Bug 439874	*S*	17-Mar-15
22315734	11-Mar-14	hadoop	1.0.0	WritableComparator.java	44	54	25-Aug-11	HADOOP-11323	*S*	20-Nov-14
23520731	7-May-14	hibernate	4.2.2	SchemaUpdate.java	115	168	22-May-13	HHH-10458	*S*	5-Feb-16
18232672	14-Aug-13	log4j	1.2.16	SMTPAppender.java	207	228	31-Mar-10	Bug 44644	*R*	18-Oct-08
17697173	17-Jul-13	lucene	4.3.0	SlowSynonymFilterFactory.java	38	52	6-Apr-13	LUCENE-4095	*D*	31-May-12
21734562	12-Feb-14	tomcat	7.0.2	FormAuthenticator.java	51	61	4-Aug-10	BZ 59823	*R*	4-Aug-16
12593810	26-Sep-12	poi	3.6	WorkbookFactory.java	49	60	7-Dec-09	57593	*R*	30-Apr-15
8037824	7-Nov-11	jasperreports	3.7.4	JRVerifier.java	1221	1240	31-May-10	N/A	*D*	20-May-11
3758110	21-Sep-10	spring	3.0.5	DefaultAnnotation HandlerMapping.java	78	92	20-Oct-10	SPR-14129	*D*	20-Jan-12
14019840	24-Dec-12	struts	2.2.1	DefaultActionMapper.java	273	288	17-Jul-10	WW-4225	*S*	18-Oct-13

* *S*: modified/added/deleted statements, *D*: file has been deleted, *R*: method has been rewritten completely

Table 11: License mapping of online clones (file-level)

Type	Qualitas	Stack Overflow (CC BY-NC-SA)	QS	EX	UD
Compatible	Apache-2	Apache-2	1		
	EPLv1	EPLv1	2		1
	Proprietary	Proprietary		2	
	Sun Microsystems	Sun Microsystems		3	
	No license	No license	20	9	2
	No license	CC BY-SA 3.0		1	
Total			23	15	3
Incompat.	AGPLv3/3+	No license	1		4
	Apache-2	No license	46	14	12
	BSD/BSD3	No license	4		1
	CDDL or GPLv2	No license			6
	EPLv1	No license	10		6
	GPLv2+/3+	No license	8	48	7
	LesserGPLv2.1+/3+	No license	16		9
	MPLv1.1	No license			1
	Oracle	No license		3	
	Proprietary	No license		1	2
	Sun Microsystems	No license		1	2
	Unknown	No license		11	
	LesserGPLv2.1+	New BSD3	1		
Total			86	78	50

Incompatible license: there are 214 clone pairs which do not contain licensing information after they are posted on Stack Overflow or contain a different license from their Qualitas clone counterparts. Almost all (85) of **QS** clone pairs have their licensing terms removed or changed when posted on Stack Overflow. One QS clone pair posted by a JFreeChart developer changed its license from Lesser GPL v2.1+ to New BSD 3-clause. The developer may intentionally changed the license to be more suitable to Stack Overflow since New BSD 3-clause license allows reuse without requiring the same license in the distributing software or statement of changes.

For **EX** clone pairs, we searched for licensing terms of the original source code from the external sources. We found that 78 out of 93 EX clone pairs have incompatible licenses. Similarly, the license statement was removed from Stack Overflow snippets.

Of 53 **UD** clone pairs, 50 pairs have incompatible licenses. Again, most clones in Qualitas contain a license while the Stack Overflow snippets do not.

For RQ4, we found 214 code snippets on Stack Overflow that could potentially violate the license of their original software. The majority of them do not contain licensing statements after they have been copied to Stack Overflow. For 164 of them, we were able to identify, with evidence, where the code snippet has been copied from.

Note that the code snippets could potentially violate the license, but do not necessarily do so. In the example where the JFreeChart developer copied his own code, she or he was free to change the license. The same may have occurred with any of the 214 code snippets.

3.5 RQ5: Stack Overflow Answerers' Awareness

We received 201 answers (33% response rate) from 607 emails we sent to the Stack Overflow answerers. The response rate was high considering other online surveys in software engineering [48]. We only present a summary of the survey answers in this paper and the full analysis is available as a research note [49].

Table 12: Experience of Stack Overflow answerers

Experience	Amount	Percent
Less than a year	1	0.5%
1 – 2 years	1	0.5%
3 – 5 years	30	14.9%
5 – 10 years	58	28.9%
More than 10 years	111	55.2%

Table 13: Frequency of including code snippets in answers

Include code snippets	Amount	Percent
Very Frequently (81–100% of the time)	84	42%
Frequently (61–80% of the time)	63	31%
Occasionally (41–60% of the time)	40	20%
Rarely (21–40% of the time)	11	6%
Very Rarely (1–20% of the time)	2	1%
Never (0% of the time)	1	1%
Total	201	100%

3.5.1 General Information

As shown in Table 12, the majority of the answerers are experienced developers with more than 10 years of experience (55.2%) or between 5 to 10 years (28.9%). They are active users and regularly answer questions. 49 participants (24%) answer questions on Stack Overflow every day.

3.5.2 Code Snippets in Answers

84 and 63 answerers include code snippets in more than 80% and 60% of their answers respectively. Interestingly, there is one answerer who never include code snippet in his/her answers (Table 13).

We asked the remaining 200 participants for the origins of code snippets in their answers. We provided six locations including the answerer's personal projects, the answerer's company projects, open source projects, writing the code from scratch, copying and modifying the code from the question, and others (e.g. code that are copied from other questions or answers on Stack Overflow) and we asked them to rate how often they copied the code from these locations. The results are shown in Figure 9. Looking at

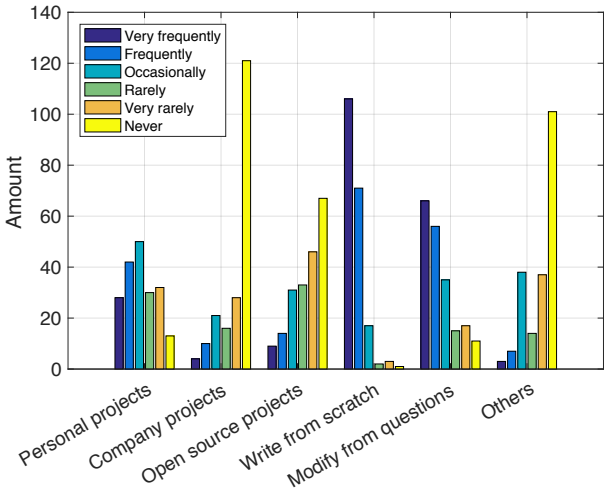


Figure 9: The sources of code snippets in Stack Overflow answers

Table 14: Notifications of outdated code snippets in answers

Notified of outdated code	Amount	Percent
Very frequently (81–100% of my answers)	2	1%
Frequently (61–80% of my answers)	1	0.5%
Occasionally (41–60% of my answers)	9	4.5%
Rarely (21–40% of my answers)	16	8%
Very rarely (1–20% of my answers)	103	51.5%
Never (0% of my answers)	69	34.5%
Total	200	100%

Table 15: Fixing of outdated code snippets in answers

Fixing of outdated code	Amount	Percent
Very frequently (81–100% of the cases)	48	36.6%
Frequently (61–80% of the cases)	27	20.6%
Occasionally (41–60% of the cases)	30	22.9%
Rarely (21–40% of the cases)	11	8.4%
Very rarely (1–20% of the cases)	8	6.1%
Never (0% of the cases)	7	5.3%
Total	131	100.0%

the Very Frequently section, we can see that the answerers mainly write new code from scratch (106) or copy and modify the code snippets from the question for each answer (66), while fewer numbers are from other sources including their personal projects (28), their company projects (4), and open source projects (9). Although copying from open source projects is not the most popular choice, the answerers still rely on them sometimes. As shown in the figure, there are 14, 31, and 33 participants who frequently, occasionally, and rarely copied code snippets from open source projects.

3.5.3 Outdated Code Snippets

How often are Stack Overflow answerers aware of the outdated code when they answer a question on Stack Overflow?

We asked the answerers if they have ever been notified about outdated code in their answers. 111 participants selected *Yes* while the other 89 participants selected *No*. However, we found inconsistent results when we asked a follow up question on the frequency of being notified. As displayed in Table 14, the number of participants who has *Never* been notified about outdated code snippets in their answers drops from 89 to 69.

We found that although the answerer have been notified of outdated code in their answers, for 51.5% of them such notifications occur very rarely (only 1–20% of the answers). Only 3 participants reported that they were notified in more than 60% of their answers.

We then asked 131 participants who have been notified of their outdated code a follow-up question *"how frequently did you fix your outdated code on Stack Overflow?"*. The answers, depicted in Table 15, show that more than half of them (57.2%) very frequently or frequently fix the outdated code snippets. However, there are 19.8% of the answerers in both groups who rarely, very rarely, or never fix their outdated code.

To answer RQ 5.1, although most of the Stack Overflow answerers are aware that their code can be outdated, 51.5% of the answerers were very rarely notified and 35.5% have

Table 16: Inclusion of software license in answer

Include license?	Amount
No.	197
Yes, in code comment	1
Yes, in text surrounding the code	2
Total	200

never been notified of outdated code in the answers. After being notified, 19.8% of them rarely or never fix the outdated code.

The responses from the answerers support our findings of outdated online code clones in RQ3. The low percentage of outdated code notifications reflect the experience of high reputation answerers who accumulate the reputation for a long time. Due to the voting mechanism of Stack Overflow, high reputation users usually provide high quality answers to earn up votes from other users. They are careful when posting code snippets in the answer to avoid problems and, vice versa, getting down votes. It would be interesting to compare the findings to Stack Overflow answerers who are newer and have lower reputation. However, we leave it to future work.

3.5.4 License of Code Snippets

How often are Stack Overflow answerers aware of licensing conflicts when they answer a question on Stack Overflow?

Among the 200 answerers who include code snippets in their answers, 124 answerers are aware that Stack Overflow apply Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) to content in the posts, including code snippets, while the rest (76) are not. Nevertheless, as shown in Table 16, almost all of them (197) reported that they did not include license statement in their code snippets due to several reasons. First, some answerers chose to post only their own code or code that was adapted from the question, hence they are automatically subjected to CC BY-SA 3.0. Second, they copied code from company or open source projects that they knew were permitted to be publicly distributed. Third, some answerers believe that code snippets in their answers are too small to claim any intellectual property on them and fall under fair use [17].

While nobody explicitly includes a software license in their snippets, many users include a statement on their profile page that all their answers are under a certain license. For example, a Stack Overflow user includes the following text in his/her profile page.

> *All code posted by me on Stack Overflow should be considered public domain without copyright. For countries where public domain is not applicable, I hereby grant everyone the right to modify, use and redistribute any code posted by me on Stack Overflow for any purpose. It is provided "as-is" without warranty of any kind.*

Many users either declare their snippets to be public domain, or they grant additional licenses, e.g. Apache 2.0 or MIT/Expat.

We asked the answerers a follow-up question of how frequently they checked for a conflict between software license of the code snippets they copied to their answers and Stack Overflow's CC BY-SA 3.0. As shown in Table 17, approximately 69% of answerers did not perform the checking. Nonetheless, there are about 10.5% of the answerers who very frequently or frequently check for licensing conflicts when they copy code snippets to Stack Overflow.

Table 17: Checking for licensing conflicts with CC BY-SA 3.0

Check license conflicts?	Amount	Percent
Very Frequently (81–100% of the time)	14	7%
Frequently (61–80% of the time)	7	3.5%
Occasionally (41–60% of the time)	10	5%
Rarely (21–40% of the time)	16	8%
Very rarely (1–20% of the time)	15	7.5%
Never (0% of the time)	138	69%
Total	200	100%

To answer RQ 5.2, 124 answerers out of 200 (62%) are aware of Stack Overflow's CC BY-SA 3.0 license applied to code snippets in questions and answers. However, only 3 answerers explicitly include software license in their answers. Some answerers choose to include the license in their profile page instead. 69% of the answerers never check for licensing conflicts between their copied code snippets and Stack Overflow's CC BY-SA 3.0.

Similar to outdated code, the responses from the answerers support our findings of licensing conflicts in online code clones in RQ4 and possibly explain why most of the clones copied from open source projects and external sources do not include their original software license.

3.5.5 Open Comments

We also invited the participants to give a free-form comment regarding their concerns of answering Stack Overflow with code snippets. Besides the one we present earlier in the introduction, these are interesting comments we received.

1) *"When I copy code it's usually short enough to be considered "fair use" but I am not a lawyer or copyright expert so some guidance from Stack Overflow would be helpful. I'd also like the ability to flag/review questions that violate these guidelines."*
2) *"My only concern, albeit minor, is that I know people blindly copy my code without even understanding what the code does."*
3) *"The main problem for me/us is outdated code, esp. as old answers have high google rank so that is what people see first, then try and fail. Thats why we're moving more and more of those examples to knowledge base and docs and rather link to those."*
4) *"Lot of the answers are from hobbyist so the quality is poor. Usually they are hacks or workarounds (even MY best answer on Stack Overflow is a workaround)."*

The comments highlight that Stack Overflow users are unsure about the legal implications of copying code, that code is copied without understanding it, and that the quality of code on Stack Overflow is often low.

3.6 Overall Discussion

Our study discovers links from code in open source projects to code snippets on Stack Overflow using clone detection

techniques. These links enable us to discover toxic code snippets with outdated code or licensing problems. The links can be exploited further to mitigate the problems of reusing outdated online clones and incompatible license on Stack Overflow code snippets. We propose the following actionable items:

- **Preventive measure:** We encourage Stack Overflow to enforce attribution when source code snippets have been copied "from" licensed software projects to Stack Overflow. Moreover, an IDE plug-in that can automatically detect pasted source code and follow the link to Stack Overflow and then to the original open source projects, could also prevent the issue of license violation.
- **Detective measure:** A system to detect outdated source code snippets on Stack Overflow is needed. The system can leverage the online clone detection techniques in this study to periodically check if the cloned snippets are still up-to-date with their originals. With such a system, the poster can be notified when the code has been updated in the original project so that he/she can update their code on Stack Overflow accordingly. On the other hand, with a crowdsourcing solution using an IDE plug-in, developers can also report the corrected version of outdated code back to the original Stack Overflow threads when they reuse outdated code and make corrections to them.

The implementation and the evaluation of the above actionable items is part of the agenda of our future work.

4 Threats to Validity

Internal validity: We applied different mechanisms to ensure the validity of the clone pairs we classified. First, we used two widely-used clone detection tools, Simian and SourcererCC. We tried five other clone detectors but could not add them to the study due to their scalability issues and susceptibility to incomplete code snippets. We adopted Bellon's agreement metric [7] to merge clone pairs for the manual classification and avoid double counting of the same clone pairs. Nevertheless, our study might still suffer from false negatives, i.e. online code clones that are not reported by the tools or are filtered out by the size (less than 10 lines) within the clone detection process. We selected accepted answers on Stack Overflow in this study to focus on code snippets that solve the question's problem and are often showed on top of the answer list. We investigated the 72,365 Stack Overflow code snippets used in our study and found that 62,245 of them (86%) are also the highest voted answers. We plan to incorporate the highest voted answers in future work.

Our seven patterns of online code cloning may not cover all possible online cloning patterns. However, instead of defining the patterns beforehand, we resorted to extract them from the data sets. We derived them from a manual investigation of 679 online clone pairs and adopted one pattern from the study by Kapser et al. [28].

The 2,302 clone pairs classified by the first and the third author are subject to manual judgement and human errors. Although we tried our best to be careful on searching for evidence and classifying the clones, some errors may still exist. We mitigated this problem by having two investigators to cross check the classifications and found 145 cases that lead to better classification results. This validation process can be even improved by employing an external investigator.

External validity: We carefully chose the data sets for our experiment so the findings could be generalise as much as possible. We selected Stack Overflow because it is one of the most popular programming Q&A websites available with approximately 7.6 million users. There are a large number of code snippets reused from the site [3] and there are also several studies encouraging of doing so (e.g. [30], [42], [45], [46]). Nonetheless, it may not be representative to all the programming Q&A websites.

Regarding the code snippets, we downloaded a full data dump and extracted Java accepted answers since they are the most likely ones to be reused. Our findings are limited to these restrictions. They may not be generalised to all programming languages and all answers on Stack Overflow. We chose the curated Qualitas corpus for Java open source projects containing 111 projects [65]. The projects span over several areas of software and has been used in several empirical studies [6], [41], [64], [67]. Although it is a curated and well-established corpus, it may not fully represent all Java open source software available.

5 Related Work

Stack Overflow is a gold mine for software engineering research and has been put to use in several previous studies. In terms of developer-assisting tools, Seahawk is an Eclipse plug-in that searches and recommends relevant code snippets from Stack Overflow [45]. A follow up work, Prompter, by Ponzanelli et al. [46] achieves the same goal but with improved algorithms. The code snippets on Stack Overflow are mostly examples or solutions to programming problems. Hence, several code search systems use whole or partial data from Stack Overflow as their code search databases [15], [30], [42], [61], [62]. Furthermore, Treude et al. [66] use machine learning techniques to extract insight sentences from Stack Overflow and use them to improve API documentation.

Another research area is knowledge extraction from Stack Overflow. Nasehi et al. [40] studied what makes a good code example by analysing answers from Stack Overflow. Similarly, Yang et al. [72] report the number of reusable code snippets on Stack Overflow across various programming languages. Wang et al. [68] use Latent Dirichlet Allocation (LDA) topic modelling to analyse questions and answers from Stack Overflow so that they can automatically categorise new questions. There are also studies trying to understand developers' behaviours on Stack Overflow, e.g. [8], [11], [39], [53].

Code clone detection is a long-standing research topic in software engineering. Whether clones are good or bad for software is still controversial [21], [23], [24], [28], [29], [35], [56]. Code clones have several applications such as software plagiarism detection [47], source code provenance [13], and software licensing conflicts [19].

Two code fragments are clones if they are similar enough according to a given definition of similarity [7]. Given an open interpretation of "definition of similarity", there are various clone detection tools and their siblings, code plagiarism detectors, invented based on plethora of different code similarity measurements [50], [51], [54], [63]. Many tools do not work on original source code directly but detect clones at an intermediate representation such as tokens [9], [16], [22], [26], [37], [47], [56], [57], [59], AST [5], [25] or program dependence graphs [32], [34].

Cloning patterns is initially defined by Kapser et al. [28], [29] by studying clones in Linux file systems and deriving 11 patterns of code cloning. Our study adopted one of the patterns into our online code cloning patterns.

Clone agreement is useful when a clone oracle is absent. By exploiting the different behaviours of clone detectors, one can look for their agreement and obtain highly-confident clones [7], [70]. Clone pairs that are agreed by multiple tools are more assured to be true clones than the ones reported by only a single tool [18], [52], [70].

Software licensing is crucial for open source and industrial software development. Di Penta et al. [14] studied the evolution of software licensing in open source software and found that licensing statements change over time. German et al. [19] found that licensing conflicts occur between the clone siblings, i.e. clones among different systems that come from the same source. Later, German et al. [20] created an automated tool for software license identification, Ninka, which is used in our online clone license analysis.

Reusing of outdated third-party source code occurs in software development. Xia et al. [71] show that a large number of open source systems reuse outdated third-party libraries from popular open source projects. Using the outdated code give detrimental effects to the software since they may introduce vulnerabilities. Our study similarly finds outdated code on Stack Overflow.

Work similar to ours are studies by An et al. [3], Abdalkareem et al. [1], and Baltes et al. [4]. An et al. investigated clones between 399 Android apps and Stack Overflow posts. They found 1,226 code snippets which were reused from 68 Android apps. They also observed that there are 1,279 cases of potential license violations. The authors rely on the timestamp to judge whether the code has been copied from/to Stack Overflow along with confirmations from six developers. Instead of Android apps, we investigated clones between Stack Overflow and 111 open source projects. Their results are similar to our findings that there exist clones from software projects to Stack Overflow with potential licensing violations. Abdalkareem et al. [1] detected clones between Stack Overflow posts and Android apps from the F-Droid repository and used timestamps to determine the direction of copying. They found 22 apps containing code cloned from Stack Overflow. They reported that cloned code is commonly used for enhancing existing code. Their analysis shows that the cloned code from Stack Overflow have negative effects to quality of the apps. The median percentage of bug fixing commits after adding Stack Overflow code (19.09%) is higher than before adding the code (5.96%) with statistical significance. Baltes et al. [4] discovered that two thirds of the clones from the 10 most frequently referenced Java code snippets on Stack Overflow do not contain attributions.

6 Conclusions

Online code clones are clones that have been copied to Q&A websites such as Stack Overflow. We classified 2,302 clone pairs using seven patterns of online code cloning. We discovered 154 clone pairs that have been copied, with evidence, from Qualitas projects to Stack Overflow, 111 clone pairs that have been copied from external sources besides Qualitas to Stack Overflow, and 65 clone pairs that are highly similar but without evidence of copying.

We performed a detailed analysis of the online clone pairs on two aspects: outdated code and potential license violation. The investigation of the 154 clone pairs copied, with evidence, from Qualitas to Stack Overflow reveals that 101 of them are outdated. Moreover, we found 214 code snippets on Stack Overflow that could potentially violate the license of their original software.

The online survey of 201 high-reputation Stack Overflow developers (i.e. answerers) show that although the developers are aware of outdated code in their answers, 19.8% of them rarely or never fix the outdated code. In addition, 62% of the participants are aware of Stack Overflow CC BY-SA 3.0 license applied to code snippets on the website. Only 3 answerers include the original license in their answers. 69% of the answerers never check for licensing conflicts between the original code and CC BY-SA 3.0 enforced by Stack Overflow.

This study is among, if not the first, to address the important issues of toxic code snippets, including outdated and license-violating online code clones, on programming Q&A websites using a hybrid methodology of automatic code clone detection and a manual clone investigation.

Acknowledgments

The authors would like to thank Prof. Cristina Lopes and Di Yang from University of California, Irvine for their help in running SourcererCC clone detector and implementing a custom tokeniser for Stack Overflow snippets.

References

[1] Rabe Abdalkareem, Emad Shihab, and Juergen Rilling. On code reuse from stackoverflow: An exploratory study on android apps. *Information and Software Technology*, 88:148–158, aug 2017.

[2] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. You get where you're looking for: The impact of information sources on code security. In *SP '16*, pages 289–305, 2016.

[3] Le An, Ons Mlouki, Foutse Khomh, and Giuliano Antoniol. Stack Overflow: A code laundering platform? In *SANER '17*, 2017.

[4] Sebastian Baltes, Richard Kiefer, and Stephan Diehl. Attribution required: Stack Overflow code snippets in GitHub projects. In *Proceedings of the 39th International Conference on Software Engineering Companion (ICSE-C'17)*, pages 161–163, 2017.

[5] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier. Clone detection using abstract syntax trees. In *ICSM'98*, pages 368–377, 1998.

[6] Nels E. Beckman, Duri Kim, and Jonathan Aldrich. An empirical study of object protocols in the wild. In *ECOOP '11*, pages 2–26, 2011.

[7] Stefan Bellon, Rainer Koschke, Giuliano Antoniol, Jens Krinke, and Ettore Merlo. Comparison and evaluation of clone detection tools. *Transactions on Software Engineering*, 33(9):577–591, 2007.

[8] Amiangshu Bosu, Christopher S. Corley, Dustin Heaton, Debarshi Chatterji, Jeffrey C. Carver, and Nicholas A. Kraft. Building reputation in StackOverflow: An empirical investigation. In *MSR '13*, pages 89–92, 2013.

[9] Steven Burrows, S. M. M. Tahaghoghi, and Justin Zobel. Efficient plagiarism detection for large code repositories. *Software: Practice and Experience*, 37(2):151–175, 2007.

[10] Wai Ting Cheung, Sukyoung Ryu, and Sunghun Kim. Development nature matters: An empirical study of code clones in JavaScript applications. *Empirical Software Engineering*, pages 517–564, 2015.

[11] Morakot Choetkiertikul, Daniel Avery, Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. Who will answer my question on Stack Overflow? In *ASWEC '15*, pages 155–164, 2015.

[12] James R. Cordy and Chanchal K. Roy. The NiCad clone detector. In *ICPC '11*, pages 3–4, 2008.

[13] Julius Davies, Daniel M. German, Michael W. Godfrey, and Abram Hindle. Software bertillonage: Determining the provenance of software development artifacts. *Empirical Software Engineering*, 18:1195–1237, 2013.

[14] Massimiliano Di Penta, Daniel M. German, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. An exploratory study of the evolution of software licensing. In *ICSE '10*, volume 1, page 145, 2010.

[15] Themistoklis Diamantopoulos and Andreas L. Symeonidis. Employing source code information to improve question-answering in Stack Overflow. In *MSR '15*, pages 454–457, 2015.

[16] Z. Duric and D. Gasevic. A source code similarity system for plagiarism detection. *The Computer Journal*, 56(1):70–86, 2012.

[17] Intellectual property for CS students: Copyrights – fair use. https://www.cs.duke.edu/courses/cps182s/fall02/cscopyright/Copyrights/Copyright-Fairuse.htm, 2017.

[18] Marco Funaro, Daniele Braga, Alessandro Campi, and Carlo Ghezzi. A hybrid approach (syntactic and textual) to clone detection. In *IWSC '10*, pages 79–80, 2010.

[19] Daniel M. German, Massimiliano Di Penta, Yann-Gael Gueheneuc, and Giuliano Antoniol. Code siblings: Technical and legal implications of copying code between applications. In *MSR '09*, pages 81–90, 2009.

[20] Daniel M. German, Yuki Manabe, and Katsuro Inoue. A sentence-matching method for automatic license identification of source code files. In *ASE '10*, page 437, 2010.

[21] Nils Göde and Jan Harder. Clone stability. In *CSMR '11*, pages 65–74, 2011.

[22] Nils Göde and Rainer Koschke. Incremental clone detection. In *CSMR '09*, pages 219–228, 2009.

[23] Jan Harder and Nils Göde. Cloned code: stable code. *Journal of Software: Evolution and Process*, 25(10):1063–1088, 2013.

[24] Keisuke Hotta, Yukiko Sano, Yoshiki Higo, and Shinji Kusumoto. Is duplicate code more frequently modified than non-duplicate code in software evolution? In *IWPSE-EVOL '10*, page 73, 2010.

[25] Lingxiao Jiang, Ghassan Misherghi, Zhendong Su, and Stephane Glondu. DECKARD: Scalable and accurate tree-based detection of code clones. In *ICSE '07*, pages 96–105, 2007.

[26] Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue. CCFinder: a multilinguistic token-based code clone detection system for large scale source code. *Transactions on Software Engineering*, 28(7):654–670, 2002.

[27] Cory Kapser and Michael Godfrey. "cloning considered harmful" considered harmful. In *WCRE '06*, pages 19–28, 2006.

[28] Cory Kapser and Michael W Godfrey. Toward a taxonomy of clones in source code: A case study. In *ELISA '03*, pages 67–78, 2003.

[29] Cory J. Kapser and Michael W. Godfrey. "cloning considered harmful" considered harmful: patterns of cloning in software. *Empirical Software Engineering*, 13(6):645–692, 2008.

[30] Iman Keivanloo, Juergen Rilling, and Ying Zou. Spotting working code examples. In *ICSE '14*, pages 664–675, 2014.

[31] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of Survey Research Part 2: Designing a Survey Sample size Experimental designs. *Software Engineering Notes*, 27(1):18–20, 2002.

[32] Raghavan Komondoor and Susan Horwitz. Using slicing to identify duplication in source code. In *SAS'01*, pages 40–56, 2001.

[33] Rainer Koschke, Raimar Falke, and Pierre Frenzel. Clone detection using abstract syntax suffix trees. In *Proceedings of the 13th Working Conference on Reverse Engineering (RE '06)*, pages 253–262, 2006.

[34] Jens Krinke. Identifying similar code with program dependence graphs. In *WCRE '01*, pages 301–309, 2001.

[35] Jens Krinke. Is cloned code more stable than non-cloned code? In *SCAM '08*, pages 57–66, 2008.

[36] Jens Krinke, Nicolas Gold, Yue Jia, and David Binkley. Cloning and copying between GNOME projects. In *MSR '10*, pages 98–101, 2010.

[37] Zhenmin Li, Shan Lu, Suvda Myagmar, and Yuanyuan Zhou. CP-Miner: Finding copy-paste and related bugs in large-scale software code. *Transactions on Software Engineering*, 32(3):176–192, 2006.

[38] Manishankar Mondal, Md. Saidur Rahman, Ripon K Saha, Chanchal K. Roy, Jens Krinke, and Kevin A Schneider. An empirical study of the impacts of clones in software maintenance. In *ICPC '11*, pages 242–245, 2011.

[39] Dana Movshovitz-Attias, Yair Movshovitz-Attias, Peter Steenkiste, and Christos Faloutsos. Analysis of the reputation system and user contributions on a question answering website: StackOverflow. In *ASONAM '13*, pages 886–893, 2013.

[40] Seyed Mehdi Nasehi, Jonathan Sillito, Frank Maurer, and Chris Burns. What makes a good code example?: A study of programming Q&A in StackOverflow. In *ICSM '12*, pages 25–34, 2012.

[41] Cyrus Omar, Young Seok Yoon, Thomas D. LaToza, and Brad A. Myers. Active code completion. In *ICSE '12*, pages 859–869, 2012.

[42] Jin-woo Park, Mu-Woong Lee, Jong-Won Roh, Seung-won Hwang, and Sunghun Kim. Surfacing code in the dark: an instant clone search approach. *Knowledge and Information Systems*, 41(3):727–759, 2014.

[43] Shari Lawrence Pfleeger and Barbara A. Kitchenham. Principles of survey research part 1: Turning lemons into lemonade. *Software Engineering Notes*, 26(6):16, 2001.

[44] Pmd's copy/paste detector (cpd). https://pmd.github.io/pmd-5.8.1/usage/cpd-usage.html, 2017.

[45] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. Seahawk: Stack Overflow in the IDE. In *ICSE '13*, pages 1295–1298, 2013.

[46] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. Mining StackOverflow to turn the IDE into a self-confident programming prompter. In *MSR '14*, pages 102–111, 2014.

[47] Lutz Prechelt, Guido Malpohl, and Michael Philippsen. Finding plagiarisms among a set of programs with JPlag. *Journal of Universal Computer Science*, 8(11):1016–1038, 2002.

[48] Teade Punter, Marcus Ciolkowski, Bernd Freimut, and Isabel John. Conducting on-line surveys in software engineering. In *Proceedings of the International Symposium on Empirical Software Engineering (ISESE'03)*, pages 80–88, 2003.

[49] Chaiyong Ragkhitwetsagul and Jens Krinke. Awareness and experience of developers to outdated and license-violating code on stack overflow: The online survey. In *UCL Computer Science Research Notes RN/17/10*, 2017.

[50] Chaiyong Ragkhitwetsagul, Jens Krinke, and David Clark. A comparison of code similarity analysers. *Empirical Software Engineering*. To appear.

[51] Chaiyong Ragkhitwetsagul, Jens Krinke, and David Clark. Similarity of source code in the presence of pervasive modifications. In *SCAM '16*, pages 117–126, 2016.

[52] Chaiyong Ragkhitwetsagul, Matheus Paixao, Manal Adham, Saheed Busari, Jens Krinke, and John H Drake. Searching for configurations in clone evaluation – a replication study. In *SSBSE '16*, 2016.

[53] Christoffer Rosen and Emad Shihab. What are mobile developers asking about? A large scale study using Stack Overflow. *Empirical Software Engineering*, 21(3):1192–1223, 2016.

[54] Chanchal K. Roy and James R. Cordy. An empirical study of function clones in open source software. In *WCRE '08*, 2008.

[55] Vaibhav Saini, Hitesh Sajnani, and Cristina Lopes. Comparing quality metrics for cloned and non cloned Java methods: A large scale empirical study. In *ICSE '16*, pages 256–266, 2016.

[56] Hitesh Sajnani, Vaibhav Saini, Jeffrey Svajlenko, Chanchal K Roy, and Cristina V Lopes. SourcererCC: Scaling code clone detection to big-code. In *ICSE '16*, pages 1157–1168, 2016.

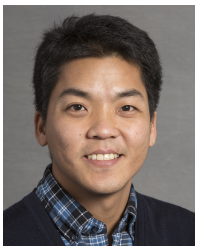
[57] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: Local algorithms for document fingerprinting. In *SIGMOD'03*, 2003.

[58] Simian–similarity analyser. http://www.harukizaemon.com/simian, 2015.

[59] Randy Smith and Susan Horwitz. Detecting and measuring similarity in code clones. In *IWSC'09*, 2009.

[60] Christopher Jon Sprigman. Oracle v. Google. *Communications of the ACM*, 58(5):27–29, 2015.

[61] Kathryn T Stolee, Sebastian Elbaum, and Daniel Dobos. Solving the search for source code. *Transactions on Software Engineering and Methodology*, 23(3):1–45, 2014.
[62] Siddharth Subramanian and Reid Holmes. Making sense of online code snippets. In *MSR '13*, pages 85–88, 2013.
[63] Jeffrey Svajlenko and Chanchal K. Roy. Evaluating modern clone detection tools. In *ICSME'14*, pages 321–330, 2014.
[64] Craig Taube-Schock, Robert J. Walker, and Ian H. Witten. Can we avoid high coupling? In *ECOOP '11*, pages 204–228, 2011.
[65] Ewan Tempero, Craig Anslow, Jens Dietrich, Ted Han, Jing Li, Markus Lumpe, Hayden Melton, and James Noble. Qualitas corpus: A curated collection of java code for empirical studies. In *APSEC '10*, pages 336–345, 2010.
[66] Christoph Treude and Martin P Robillard. Augmenting API documentation with insights from Stack Overflow. In *ICSE '16*, pages 392–403, 2016.
[67] Bogdan Vasilescu, Alexander Serebrenik, and Mark van den Brand. You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics. In *ICSM '11*, pages 313–322, 2011.
[68] Shaowei Wang, David Lo, and Lingxiao Jiang. An empirical study on developer interactions in StackOverflow. In *SAC '13*, pages 1019–1024, 2013.
[69] Shaowei Wang, David Lo, Bogdan Vasilescu, and Alexander Serebrenik. EnTagRec: An enhanced tag recommendation system for software information sites. In *ICSME '14*, pages 291–300, 2014.
[70] Tiantian Wang, Mark Harman, Yue Jia, and Jens Krinke. Searching for better configurations: A rigorous approach to clone evaluation. In *FSE '13*, pages 455–465, 2013.
[71] Pei Xia, Makoto Matsushita, Norihiro Yoshida, and Katsuro Inoue. Studying reuse of out-dated third-party code. *Information and Media Technologies*, 9(2):155–161, 2014.
[72] Di Yang, Aftab Hussain, and Cristina Videira Lopes. From query to usable code: An analysis of Stack Overflow code snippets. In *MSR '16*, pages 391–402, 2016.
[73] Di Yang, Pedro Martins, Vaibhav Saini, and Cristina Lopes. Stack overflow in github: Any snippets there? In *Proceedings of the International Conference on Mining Software Repositories (MSR '17)*, 2017.



Chaiyong Ragkhitwetsagul is working toward the PhD degree in Computer Science at University College London, where he is part of the Centre for Research on Evolution, Search, and Testing (CREST) and Software Systems Engineering (SSE) Group. He is supervised by Dr. Jens Krinke and Dr. David Clark. His research interests include code search, code clone detection, software plagiarism, search-based software engineering, and mining software repository.

Jens Krinke is Senior Lecturer in the Software Systems Engineering Group at the University College London, where he is Director of CREST, the Centre for Research on Evolution, Search, and Testing. His main focus is software analysis for software engineering purposes. His current research interests include software similarity, modern code review, and mutation testing. He is well known for his work on program slicing and clone detection.

Matheus Paixao is currently a PhD candidate in the Computer Science Department at University College London, where he is part of the CREST centre. He is supervised by Dr. Jens Krinke, Prof. Mark Harman, Dr. Emmanuel Letier and Dr. Yuanyuan Zhang. His research interests include software architecture, search based software engineering, technical debt, empirical software engineering and modern code review.

Giuseppe Bianco received his bachelor degree in Computer Science from the University of Molise (Italy) under the supervision of Dr. Rocco Oliveto. As part of an Erasmus+ traineeship, he spent three month at the CREST centre, University College London, UK, under the supervision of Dr. Jens Krinke. He is now a project manager at Corsi.it.

Rocco Oliveto is Associate Professor at University of Molise (Italy), where he is also the Director of the Computer Science Bachelor and Master programs and the Director of the Software and Knowledge Engineering Lab (STAKE Lab). He co-authored over 100 papers on topics related to software traceability, software maintenance and evolution, search-based software engineering, and empirical software engineering. His activities span various international software engineering research communities. He has served as organizing and program committee member of several international conferences in the field of software engineering. He was program co-chair of ICPC 2015, TEFSE 2009 and 2015, SCAM 2014, WCRE 2012, and 2013. He is general chair for SANER 2018 that will be held in Campobasso from March 20, 2018 to March 23, 2018. He is also one of the co-founders and CEO of DataSound, a spin-off of the University of Molise aiming at efficiently exploiting the priceless heritage that can be extracted from big data analysis via machine learning. More info at https://dibt.unimol.it/staff/oliveto.