

How Do Developers Utilize Source Code from Stack Overflow?

Yuhao Wu · Shaowei Wang · Cor-Paul
Bezemer · Katsuro Inoue

Received: date / Accepted: date

Abstract Technical question and answer Q&A platforms, such as Stack Overflow, provide a platform for users to ask and answer questions about a wide variety of programming topics. These platforms accumulate a large amount of knowledge, including hundreds of thousands lines of source code. Developers can benefit from the source code that is attached to the questions and answers on Q&A platforms by copying or learning from (parts of) it. By understanding how developers utilize the source code from Q&A platforms, we can provide insights for researchers which help improve next-generation Q&A platforms to help developers to reuse the source code fast and easily.

In this paper, we first conduct an exploratory study on 95 JavaScript files from 89 projects which contain source code that contains an explicit reference to a Stack Overflow post. In 49.4% of the studied cases, developers need to modify source code to make it work in the required context, with the degree of modification varying from renaming variables to rewriting the whole algorithm. Developers sometimes choose to implement an algorithm from scratch based on the descriptions from Stack Overflow answers, even if there is an implementation readily available in the post. To further understand the barriers that prevent code reuse from developers and obtain suggestions for improving the code reuse from Q&A platforms, we conduct a survey on 453 developers on Stack Overflow. We find that the top 3 barriers that prevent developers from copying code from Stack Overflow are: (1) too much code modification required to fit in their projects, (2) incomprehensive code, and (3) low code quality.

Yuhao Wu · Katsuro Inoue
Graduate School of Information Science and Technology, Osaka University, Japan
E-mail: {wuyuhao,inoue}@ist.osaka-u.ac.jp

Shaowei Wang (✉) · Cor-Paul Bezemer
SAIL, Queen's University, Canada
E-mail: {shaowei,bezemer}@cs.queensu.ca

We also summarize their suggestions for next-generation Q&A platforms and discuss implications of our study on future research along the following dimensions: (1) code quality, (2) information enhancement & management, (3) data organization, (4) license, and (5) human factor.

1 Introduction

Technical question and answer (Q&A) platforms such as Stack Overflow have become more and more important for software developers to share knowledge. Developers can post questions on these Q&A platforms, which in turn are answered by other developers. These answers often contain source code snippets.

As of August 2017, Stack Exchange reports that there are approximately 7.6 million users and 14 million questions with 23 million answers on Stack Overflow (Stack Exchange, 2017). Among those answers, 15 million (75%) have at least one source code snippet attached, which forms a huge code base for developers to reuse source code from.

However, reusing source code is not easy (Gamma *et al.*, 1995). There are two major challenges when reusing source code: (1) It is difficult for developers to find suitable source code based on their particular needs, such as language, functionality, and performance (Wang *et al.*, 2014a). To address this challenge, a number of studies have been done to help developers to locate more relevant source code snippets (Ponzanelli *et al.*, 2014a,b; Wang *et al.*, 2014a). (2) It may require additional effort to integrate the found source code work in developers' own context. For example, parameters may need to be adjusted, or additional source code may need to be added (Cottrell *et al.*, 2008). To address this challenge, prior studies proposed various techniques (Cottrell *et al.*, 2008; Meng *et al.*, 2011; Yellin and Strom, 1997; Meng *et al.*, 2013; Hua *et al.*, 2015), such as variable renaming based on the particular context.

Prior studies focus on helping developers locate the source code from online resources (e.g., Q&A platforms) then integrate that source code into their own project. However, little is known about how developers utilize these source code from Q&A platforms into their own project in practice. Such knowledge will help us understand what the potential barriers that prevent developers from reusing code from Q&A platforms are. By understanding how developers utilize code from Q&A platforms, we can get insights to facilitate developers reusing the source code from Q&A platforms and future research in next-generation Q&A platforms.

In this paper, we first conduct an exploratory study on 95 JavaScript files from 89 projects on GitHub, which contain at least one link to a Stack Overflow post. We manually study each source file and its linked Stack Overflow post, to investigate how developers reuse code from Stack Overflow. In particular, we focus on the following research questions:

RQ1: How often and to what extent do developers need to modify source code from Stack Overflow in order to make it work in their own projects?

78.2% of the reused source code is modified from simple refactoring to a complete reimplementation, which supports the prior studies on automatic code integration (Feldthaus and Møller, 2013; Alnusair *et al.*, 2016; Wang *et al.*, 2016b). In 24.2% of the cases, developers reimplement code based on the idea of a Stack Overflow answer, which suggests that Q&A platforms should consider to summarize key points that are discussed in a post to give developers a quick overview of the question and its answers.

RQ2: *From which part of the Stack Overflow post does the reused source code come?*

Developers reuse source code from non-accepted answers (26%) for different reasons, e.g., the simplicity and performance of the source code. Some developers even adopt answers that are total opposites from what the original asker wanted but meet their needs. Hence, Q&A platforms should consider to improve the way of organizing answers, so that developers can find the most suitable answers based on their requirements easily, such as voting on the different aspects (e.g., readability, performance) of answers or adding tags for answers.

To further understand the barriers that may prevent code reuse and to collect suggestions for improving the code reuse process on Q&A platforms, we conduct a survey on 453 developers from Stack Overflow. In our survey, we focus on the following research questions:

RQ3: *What are the preferences of developers when it comes to reusing code?* Participants reimplement source code slightly more frequently than reuse source code from Q&A platforms. Code modification according to their projects, code comprehension, and code quality are the top reasons that participants gave regarding why they prefer reimplementing over reusing source code. Our findings provide empirical evidence to support current research on code integration and code comprehension, and highlight the need of providing code quality indicators on next-generation Q&A platforms.

RQ4: *What is the influence of the code license of Stack Overflow?* In the exploratory study, we noticed that the number of links to Stack Overflow posts in the source code of open source projects is low, which raises the question whether developers are aware of the license requirements of Q&A platforms, since attribution is required by some Q&A platforms such as Stack Overflow. The result shows that 80% of the participants do not have a good understanding of the licenses of the Q&A platforms and most of the participants (57%) think that having more information on license is important. These findings suggest that next-generation Q&A platforms should make code licensing information more visible to developers.

RQ5: *How can code reuse be improved in next-generation Q&A platforms?* The suggestions for next-generation Q&A platforms are primarily summarized in five categories: code quality, information enhancement & management, data organization, license, and human factor. Among code quality, 42.2% of all participants suggest to integrate an online code validator and 29.7% of the participants suggest to develop a mechanism to deal with outdated source code.

The rest of this paper is organized as follows. Section 2 introduces the background and motivation of our study. Section 3 describes our data collection process. Our exploratory study is described in Section 4. Section 5 presents the survey design and results. Section 6 discusses the implications of our study on future research for next-generation Q&A platforms. Section 8 describes the threats to validity. And finally, Section 9 concludes the paper.

2 Background

In this section, we discuss the background of this study.

2.1 Question and Answer on Stack Overflow

Nowadays, technical Q&A platforms, with Stack Overflow being the most prominent, have become an important way for researchers and practitioners to obtain knowledge about and find solutions to their programming problems (Treude *et al.*, 2011; Abdalkareem *et al.*, 2017). Developers are allowed to post questions, answer questions, vote questions or answers on the Q&A platforms. In the questions or answers, developers often attach pieces of source code to explain their questions or answers. For example, Figure 1 presents an example of an answer that contains source code in Stack Overflow. The asker asked how to get the HTML of a selected object with jQuery. The answerer posted an answer that provides a solution in the form of the attached source code.

Stack Overflow allows askers to mark at most one answer as the “accepted answer” to indicate whether the answer meets his/her requirement (see Figure 1). Stack Overflow allows developers to up vote or down vote a post (e.g., a question or an answer) to express that whether the post is useful. The total up and down votes that a post receives is presented as a score beside the post. For example, the score of the question in Figure 1 is 673. In general, high-voted answer and question are usually regarded as high-quality one.

2.2 Source Code Reuse from Stack Overflow

Source code reuse from Stack Overflow can be commonly observed (An *et al.*, 2017). Developers may copy-and-paste source code from Stack Overflow posts into their own projects as long as they adhere to the Creative Commons Attribute-ShareAlike (CC BY-SA 3.0) license¹, according to an official Stack Overflow blog post (Atwood, 2009). One of the requirements of this license is that “*attribution*” is needed from the developers by putting a hyper link to the original Stack Overflow post in their source code comments. Figure 2 shows an example of code reuse. This source code snippet, which is taken from a GitHub project², is reused from the Stack Overflow post shown in Figure 1.

¹ <https://creativecommons.org/licenses/by-sa/3.0/>

² <https://goo.gl/X84SF i>


Get selected element's outer HTML


▲ 673 I'm trying to get the HTML of a selected object with jQuery. I am aware of the `.html()` function; the issue is that I need the HTML including the selected object (a table row in this case, where `.html()` only returns the cells inside the row).

▼ 122 I've searched around and found a few very 'hackish' type methods of cloning an object, adding it to a newly created div, etc, etc, but this seems really dirty. Is there any better way, or does the new version of jQuery (1.4.2) offer any kind of `outerHTML` functionality?

jquery

share edit

edited Jul 19 '11 at 7:34  Paul D. Waite 56.2k ● 41 ● 154 ● 241

asked Mar 10 '10 at 19:09  Ryan 6,572 ● 14 ● 50 ● 75

26 Answers active oldest votes

▲ 162 *2014 Edit: The question and this reply are from 2010. At the time, no better solution was widely available. Now, many of the other replies are better: Eric Hu's, or Re Capcha's for example.*

▼ This site seems to have a solution for you: [jQuery: outerHTML | Yelotofu](#)

✓

```
jQuery.fn.outerHTML = function(s) {
  return s
    ? this.before(s).remove()
    : jQuery("<p>").append(this.eq(0).clone()).html();
};
```

share edit

edited Dec 13 '14 at 18:27


answered Mar 10 '10 at 19:26  David V. 4,419 ● 2 ● 18 ● 23

Fig. 1: An example of a question and its accepted answer on Stack Overflow.

```
1 //http://stackoverflow.com/questions/2419749/get-
2 // selected-elements-outer-html
3 jQuery.fn.outerHTML = function (s) {
4   return s
5     ? this.before(s).remove()
6     : jQuery("<p>").append(this.eq(0).clone()).html();
7 };
```

Fig. 2: Source code reuse example.

3 Data Collection

The steps of our data collection and analysis processes of the first part of our study are shown in Figure 3. First, we collect source files that contain an explicit reference to a Stack Overflow post from `searchcode.com` (Searchcode, 2016b), a source code search website. Then, we remove irrelevant files such as false positives and duplicate files. Finally, we analyze the source files and the corresponding Stack Overflow posts to answer our research questions. We elaborate in more detail on our data collection and analysis processes in the

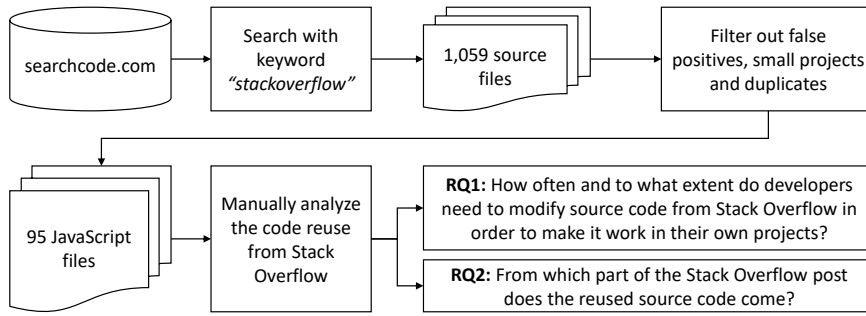


Fig. 3: An overview of our data collection and analysis processes of the exploratory study.

remainder of this section. Note that we explain the second part of our study, i.e., the survey, in Section 5.

3.1 Collecting Source Files from Open Source Project Repositories

As explained in Section 2.2, developers must cite a Stack Overflow post when they reuse code or ideas from that post. Hence, to obtain source files that contain source code snippets that are reused from Stack Overflow posts (either questions or answers), we search for source files that contain at least one hyperlink to a Stack Overflow post.

To search for such source files, we use `searchcode.com` (Searchcode, 2016b) as our search engine. `searchcode` has indexed over 20 billion lines of source code from 7 million projects. With its API (Searchcode, 2016a), we were able to collect 1,059 files in total using “*stackoverflow*” as the search keywords. In this research, we limit our search results to projects on GitHub, since GitHub has the largest project population compared to other repositories. We focus on files that are written in JavaScript since JavaScript is the most popular language on Stack Overflow (Stack Overflow, 2016).

3.2 Removing Irrelevant Files

Using “*stackoverflow*” as the search keyword can result in some source files that may contain the “*stackoverflow*” keyword without containing a link to a post, e.g., in an API name. We manually remove these false positives in this step. To mitigate the effects from small projects (Kalliamvakou *et al.*, 2014), we eliminate files that belong to projects with less than 1,000 commits and 10 contributors. We then remove the files that are duplicates of each other (e.g., files from forked projects). Finally, we ended up with 95 unique JavaScript files,

which belong to 89 projects. Within these files, 115 Stack Overflow hyperlinks were found.

4 Part I: Exploratory Study

In this section, we present and discuss the results of our exploratory study. For each research question in our exploratory study we present the motivation, approach, and results.

4.1 *RQ1: How often and to what extent do developers need to modify source code from Stack Overflow in order to make it work in their own projects?*

Motivation: Prior research has proposed several ways to utilize the source code on Stack Overflow (Rigby and Robillard, 2013; Ponzanelli *et al.*, 2013, 2014a,b). For example, Ponzanelli *et al.* (2013) presented an approach to automatically construct queries from the current context in the IDE (i.e., Eclipse) and retrieve relevant code and its corresponding discussions from Stack Overflow. However, there is no empirical evidence about the process of how developers are reusing the source code, e.g., whether they copy-and-paste the original source code without any modification or they modify it considerably. Knowing how developers reuse source code from Q&A platforms will give us insights on how to make source code reuse easier in next-generation Q&A platforms.

Approach: To understand how developers utilize source code from Stack Overflow, we manually analyze the collected JavaScript source code and the linked Stack Overflow posts. We manually extract and categorize the type of code utilization from Stack Overflow posts for each collected source file. We perform an interactive process similar to *Open Coding* (Seaman, 1999; Seaman *et al.*, 2008) for identifying the type of the code utilization. This process involves 3 phases and is performed by 3 persons (i.e., P1, P2, P3) who are the first three authors of this study as follows:

- Phase I: P1 extracts a draft list of types of source code utilization from Stack Overflow based on 20 source files and the linked Stack Overflow post. Then, P1 and P2 use the draft list to categorize the same source file collaboratively, during which the types are revised and refined. At the end of this phase, we obtain 5 types of source code utilization.
- Phase II: P1 and P2 apply the resulting types of Phase I to independently categorize all collected source file. They are instructed to take notes regarding the deficiency and ambiguity of the types for categorizing certain source files.
- Phase III: P1, P2, and P3 discuss the coding results obtained in Phase II to resolve the disagreements until a consensus is reached.

Table 1: Types of source code utilization from Stack Overflow in JavaScript files.

ID	Name	Definition	Count	Perc.
C1	Exact Copy	Developers copy-and-paste the source code from Stack Overflow without any modification .	19	21.8%
C2	Cosmetic Modification	Developers copy-and-paste the source code from Stack Overflow with modifications which do not alter the functionality of that source code (e.g., renaming identifier names to make it more readable).	8	9.2%
C3	Non-cosmetic Modification	Developers copy-and-paste the source code from Stack Overflow with modifications which alter the functionality of the source code (e.g., adding arguments to a function prototype).	35	40.2%
C4	Converting Ideas	Developers do not copy-and-paste any source code from Stack Overflow. Instead, they write the source code from scratch by applying the ideas in the answers.	21	24.2%
C5	Providing Information	Developers do not reuse any source code from Stack Overflow. Instead, they treat the Stack Overflow post as an information source related to the issue they are addressing.	4	4.6%

Table 1 shows final categorization of the types of source code utilization from Stack Overflow in JavaScript files. In our study, one pair could only be categorized as one type.

Results: 49.4% of the reused source code is modified in one way or another. Table 1 shows that 21.8% of the studied files reuse source code without modification (C1). 49.4% (C2 and C3) of the files require modification when developers reuse source code. Type C4 (24.2%) indicates that it is not exceptional that developers convert the ideas written in natural language to source code from scratch. Type C5 (4.6%) indicates that there exist developers who use Stack Overflow as a “programming manual”.

In 9.2% of the studied files, developers make cosmetic modifications when reusing source code, which may improve the readability or simplicity of the source code. In the Cosmetic Modification category, developers copy-and-paste the source code from a Stack Overflow post and make modifications to the source code which may not be necessary to make the source code work in the target project. In the example shown in Figure 4, the developers copied two lines of source code in the accepted answer from the Stack Overflow post (see Figure 4b) and renamed the variable name from `re` to `emailRegExp` (see Figure 4a).

³ <https://goo.gl/9ouSz1>

⁴ <https://goo.gl/74oVBu>

```

1 // http://stackoverflow.com/questions/46155/validate-
2 // email-address-in-javascript
3 var emailRegExp = /[a long regex string]/;
4 return emailRegExp.test(value);

```

(a) Source code from the project³.

```

1 var re = /[a long regex string]/;
2 return re.test(value);

```

(b) Source code from the Stack Overflow answer⁴.

Fig. 4: C2: An example of code reuse with cosmetic modifications.

In 24.2% of the files, developers write the source code from scratch based on the descriptions of the algorithm. In the example shown in Figure 5 and Figure 6, developers implement a function to detect whether a line intersects with the rectangle (see Figure 5), based on an answer from the Stack Overflow post shown in Figure 6.

Another example is shown in Figure 7, where the developers wrote a regular expression that extracts all Youtube video ids in a string (see Figure 7a). This source code snippet is modified based on the source code from the Stack Overflow post shown in Figure 7b, which is written in PHP. In this example, developers actually rewrote the regular expression in JavaScript based on the PHP source code from the Stack Overflow post. We categorized this file under the Converting Ideas type since developers cannot reuse the source code directly from another language, instead, they have to convert the idea and rewrite it from scratch.

Developers use Stack Overflow posts in 4.6% of the files as an information source for later reference. In 4.6% of the files, developers do not reuse any source code from Stack Overflow. Instead, they put a Stack Overflow hyperlink in their source code to provide background information about the issue or solution. For example, there is a file⁵ in which the developers have tried a solution on Stack Overflow which does not work. They then put the link to the Stack Overflow post in the source code with a comment “*Also tried a method from Stack Overflow that caused a security error in all browsers*”.

In summary, most of the source code reuse (49.4%) requires additional modification, which implies that finding the code is only the first step for code reuse, more effort is needed to facilitate code reuse from Q&A platforms after retrieving relevant code from Q&A platforms, such as adjusting the source code to the required context. Prior studies tried to integrate the source code

⁵ <https://goo.gl/4ezMUr>

⁶ <https://goo.gl/1Wn9vF>

⁷ <https://goo.gl/HK5kyV>

⁸ <https://goo.gl/eq1Dnk>

⁹ <https://goo.gl/wRoLrn>

```

1 Rect.prototype.collideLine = function(p1, p2) {
2   var x1 = p1[0];
3   var y1 = p1[1];
4   var x2 = p2[0];
5   var y2 = p2[1];
6
7   function linePosition(point) {
8     var x = point[0];
9     var y = point[1];
10    return (y2-y1)*x + (x1-x2)*y + (x2*y1-x1*y2);
11  }
12
13  var relPoses = [[this.left, this.top],
14                 [this.left, this.bottom],
15                 [this.right, this.top],
16                 [this.right, this.bottom]
17                ].map(linePosition);
18
19  var noNegative = true;
20  var noPositive = true;
21  var noZero = true;
22  relPoses.forEach(function(relPos) {
23    if (relPos > 0) {
24      noPositive = false;
25    } else if (relPos < 0) {
26      noNegative = false;
27    } else if (relPos === 0) {
28      noZero = false;
29    }
30  }, this);
31
32  if ( (noNegative || noPositive) && noZero) {
33    return false;
34  }
35  return !((x1 > this.right && x2 > this.right) ||
36          (x1 < this.left && x2 < this.left) ||
37          (y1 < this.top && y2 < this.top) ||
38          (y1 > this.bottom && y2 > this.bottom)
39          );
40 };

```

Fig. 5: C5: An example of converting descriptions into source code: source code from the project⁵ is implemented based on the description of the algorithm from Stack Overflow (see Figure 6).

to a target context automatically (Feldthaus and Møller, 2013; Alnusair *et al.*, 2016; Wang *et al.*, 2016b). Our findings also support these existing studies, and it would be interesting to integrate these code integration techniques into Q&A platforms. In addition, a number of files show that developers write the source code from scratch based on the description of an algorithm instead of reusing source code. Such files imply that Q&A platforms should consider to

```

1 Let the segment endpoints be p1=(x1 y1) and p2=(x2 y2).
2 Let the rectangle's corners be (xBL yBL) and (xTR yTR).
3
4 Then all you have to do is
5
6 A. Check if all four corners of the rectangle are on the
7 same side of the line. The implicit equation for a line
8 through p1 and p2 is:
9
10  $F(x y) = (y2-y1)x + (x1-x2)y + (x2*y1-x1*y2)$ 
11
12 If  $F(x y) = 0$ , (x y) is ON the line.
13 If  $F(x y) > 0$ , (x y) is "above" the line.
14 If  $F(x y) < 0$ , (x y) is "below" the line.
15
16 Substitute all four corners into  $F(x y)$ . If they're all
17 negative or all positive, there is no intersection. If
18 some are positive and some negative, go to step B.
19
20 B. Project the endpoint onto the x axis, and check if the
21 segment's shadow intersects the polygon's shadow. Repeat
22 on the y axis:
23
24 If  $(x1 > xTR \text{ and } x2 > xTR)$ , no intersection (line is to
25 right of rectangle).
26 If  $(x1 < xBL \text{ and } x2 < xBL)$ , no intersection (line is to
27 left of rectangle).
28 If  $(y1 > yTR \text{ and } y2 > yTR)$ , no intersection (line is
29 above rectangle).
30 If  $(y1 < yBL \text{ and } y2 < yBL)$ , no intersection (line is
31 below rectangle).
32 else, there is an intersection. Do Cohen-Sutherland or
33 whatever code was mentioned in the other answers to
34 your question.
35
36 You can, of course, do B first, then A.

```

Fig. 6: Description of the algorithm in the Stack Overflow answer⁶.

provide a summary of key points that are discussed in a Q&A post to give developers an overview of the question and its answers.

49.4% of the reused source code requires additional modification, which supports the prior studies on automatic code integration. In 24.2% of the files, developers reimplement code based on an idea, which suggests that Q&A platforms should consider to summarize key points that are discussed in a post to give developers a quick view of the question and its answers.

```

1  YOUTUBE_REGEXP: new RegExp(
2  '(?:https?://)?' + // Optional scheme. Either...
3  '(?:www\\.)?' + // Optional www subdomain
4  '(?:' + // Group host alternatives
5  'youtu\\.be/' + // Eitheryoutu.be,
6  [...])
7  ')' // End negative lookahead assertion.
8  ),

```

(a) Source code from the project in JavaScript⁷.

```

1  //Linkify youtube URLs which are not already links
2  function linkifyYouTubeURLs($text) {
3      $text = preg_replace('~(?:js|YouTubeId|Rev:...
4      # Match non-linked youtube URL in the wild...
5      https?:// # Required scheme...
6     (?:[0-9A-Z-]+\.)? # Optional subdomain.
7     (?: # Group host alternatives.
8      youtu\\.be/ # Eitheryoutu.be,
9      [...])
10     $text);
11     return $text;
12 }

```

(b) Source code from the Stack Overflow answer in PHP⁸.

Fig. 7: An example of cross-language source code reuse. Regular expression strings with the same functionality are highlighted with red.

4.2 RQ2: From which part of the Stack Overflow post does the reused source code come?

Motivation: Intuitively, we may expect that accepted or high-voted answers are the most useful. However, in RQ1 we observed that developers also reuse source code from answers that were not the highest-voted nor accepted. By understanding why developers choose the non-accepted or non-highest-voted answers, we could provide insights to help Q&A platforms organize their answers better so that developers can find solutions more easily.

Approach: We manually inspect from which part (e.g., accepted answer, non-accepted answer, or question) of the Stack Overflow post the reused source code originates. We also check whether the answer is the highest-voted one. Two of the authors manually examined each source code file and the linked post (including the question, all answers, and all comments to the answers) individually and categorized it. Discrepancies were discussed until a consensus was reached.

Results: In 26% of the files developers choose the non-accepted answers and in 63%, such non-accepted answers are not the highest-voted. The results of the categorization are shown in Table 2. As we can see

Table 2: Where does the reused source code come from?

Source	Highest-voted	Non-highest-voted	Total	Perc.
Accepted Answer	48	2	50	43%
Non-Accepted Answer	11	19	30	26%
Question	-	-	1	1%
NOT REUSE	-	-	34	30%
Total	-	-	115	100%

Listing 1: Source code in the project that implements a method to generate GUIDs.¹⁰

```

1 // http://stackoverflow.com/questions/105034/how-to-
2 // create-a-guid-uuid-in-javascript
3 function generateID() {
4     return "avalon"
5     + Math.random().toString(36).substring(2, 15)
6     + Math.random().toString(36).substring(2, 15)
7 }

```

from the results, not all source code is reused from accepted answers. In 43% of the files, developers choose source code from the accepted answers. However, there are still a considerable number (26%) of files where developers choose the source code from non-accepted answers. Moreover, among such non-accepted answers, 63% are not the highest-voted ones, which indicates that developers do not always choose source code from the accepted or highest-voted answer. In the remainder of this section, we discuss different situations in which developers reuse source code from non-accepted answers in more detail.

4.2.1 Simpler Source Code

Description: Developers choose solutions from non-accepted answers because the source code is simpler.

Example: A developer wants to implement a method to generate GUIDs. The source code in this example is shown in Listing 1.

This source code snippet is actually from a Stack Overflow non-accepted answer¹² which has 37 votes, while the accepted answer has 1290 votes. The source code provided by the accepted answer is shown in Listing 2.

The former source code in Listing 1 is quite straightforward and has higher readability. According to the description in the answer, this method in Listing 1 is not compliant with the RFC 4122 standard but has a very good performance. The author of this answer also attached a performance test result of different

¹⁰ <https://goo.gl/Z1pRMS>

¹¹ <https://goo.gl/xpAcga>

¹² <https://goo.gl/aC4auZ>

Listing 2: Source code in the accepted answer on Stack Overflow.¹¹

```

1 function guid() {
2   function s4() {
3     return Math.floor((1 + Math.random())
4       * 0x10000).toString(16)
5       .substring(1);
6   }
7   return s4() + s4() + '-' + s4() + '-' + s4()
8     + '-' + s4() + '-' + s4() + s4() + s4();
9 }

```

algorithms that are mentioned in other answers of the Stack Overflow post, which shows that this algorithm outperforms others.

Our perception is that the developer who adopted this low-voted answer prioritize performance and readability more than other metrics such as whether the generated result is compliant with a standard.

4.2.2 Fixing Bugs

Description: Source code that fixes potential bugs of the accepted answer may also get adopted by developers.

Example: A developer is looking for a method to draw a dashed line around selection area in JavaScript¹³.

The non-accepted answer¹⁴ improves the accepted answer by utilizing built-in transformation functionality of Canvas, and also handles special cases where the line is vertical, which was not addressed in the accepted answer.

4.2.3 Improving Speed

Description: Answers with higher performance will also attract developers reusing the source code.

Example: A developer is looking for an algorithm that sorts an array by the Levenshtein Distance in JavaScript. According to the comments under the accepted answer, the implementation in the accepted answer performs better than the one provided by the original asker. However, a non-accepted answer provides an improved version of the accepted answer which was described as “*Most speed was gained by eliminating some array usages*”, which was reused by the developers in their project. Thus we believe these developers give performance a higher priority.

Based on our observations, we find that different developers have different requirements for their solutions. Even if answers that are provided in the post

¹³ <https://goo.gl/gzMCGy>

¹⁴ <https://goo.gl/8foVXq>

do not meet the requirements of the asker, other developers may find them useful (e.g., solution with higher performance). For developers who are looking for solutions on Stack Overflow, it is better to go through all the answers of a relevant question instead of focusing on the accepted answers. Q&A platforms should improve the way of organizing answers, so that developers can find the most suitable answers based on their different requirements faster. For example, Q&A platforms may allow users to vote on different aspects, such as the readability or performance of the source code in an answer. The results from our user survey support the need for this improvement (see Section 6.2).

Developers reuse non-accepted or non-highest-voted answers for different reasons, such as the simplicity and performance of the source code. Some even reuse answers that are total opposites from what the asker wanted. Hence, Q&A platforms should improve the way of organizing answers, so that developers can find the most suitable answers based on their requirements easily.

5 Part II: Survey of Code Reuse From Stack Overflow

In RQ1, we observed that there are a considerable number of cases in which developers had to modify the source code before they could reuse it. To further understand how developers reuse source code from Q&A platforms, we conducted a survey amongst 453 developers. By better understanding the process of code reuse, we can make suggestions to make code reuse more efficient on next-generation Q&A platforms.

5.1 Research Questions

The goal of our survey is to gain insights on the challenges that developers face when reusing source code from Q&A platforms. In our survey, we focus on eliciting information that helps us answer the following research questions:

5.1.1 RQ3: Do developers prefer reusing or reimplementing source code?

Intuitively, reusing source code will consume less effort than reimplementing, especially if the source code is well tested. From Section 4, we observed several cases in which developers preferred reimplementing over reusing. In this part of the survey, we focus on understanding which factors make developers prefer reimplementing source code rather than reusing it.

5.1.2 RQ4: What is the influence of the code license of Stack Overflow?

Prior study shows that the amount of code reuse from Q&A platforms (i.e., Stack Overflow) is low (1.70%) (Abdalkareem *et al.*, 2017). One possible reason is that developers might not be aware of the license requirements of Q&A platforms, since attribution is required by some Q&A platforms, such as Stack

Overflow. Another possible reason is that some Q&A platforms (e.g., Stack Overflow) have a relatively restrictive license, which might hinder the developers from reusing source code from these Q&A platforms. Hence, in this RQ, it is interesting to understand whether, in the eyes of developers, license is a factor that they consider as a barrier when reusing code from Q&A platforms. In addition, previous study has observed potential license violation cases where developers reused source code from or to Q&A platforms (An *et al.*, 2017). We are also interested in studying whether having more license information is necessary in the eyes of the developers.

5.1.3 RQ5: How can code reuse be improved in next generation Q&A platforms?

The purpose of this part of the survey is to collect suggestions from the developers that can help improve next generation Q&A platforms. The goal of this RQ is to define a roadmap for future research and the implementation of next-generation Q&A platforms.

5.2 Survey Design

Two of the authors posited the survey questions that cover all the three research questions. The third author checked the questions to eliminate any ambiguity of the wording of the survey.

The details of the survey are available in the Appendix. The survey is divided into three parts:

1. Demography (Q1 - Q7): these questions collect the software engineering background of the participants.
2. Barriers (Q8 - Q17): these questions collect information about the barriers that the participants are facing in the process of reusing source code from Q&A platforms. We collected responses from participants who have ever reused source code from Q&A platforms (i.e., those who answered *yes* to Q7, which is 380 (83.9%) of all the participants).
3. Suggestions (Q18 - Q19): these questions collect suggestions for next generation Q&A platforms. Every participant can answer these two questions no matter whether they have reused source code from Q&A platforms or not.

5.3 Data Collection

We use the dataset provided by Vasilescu *et al.* (2013) to get the candidate participants. This dataset includes 93,771 email addresses of the intersection of users of GitHub and Stack Overflow. We took a random sample of 6,000 users from this dataset and sent them email invitations for our online survey. 1,935 of the emails did not reach the survey candidates because the email address

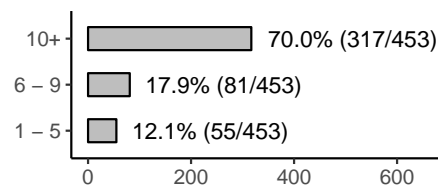


Fig. 8: Distribution of the software engineering experience of the participants in years.

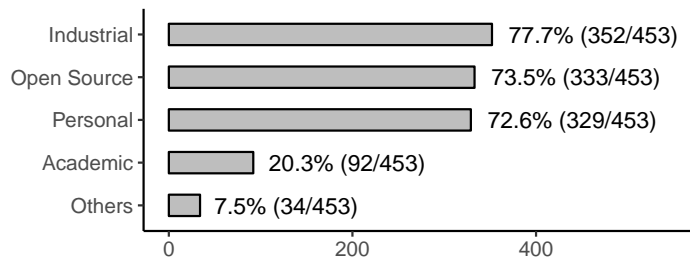


Fig. 9: Distribution of the project types of the participants.

does not exist any more. In the end, we received 453 responses which equals a response rate of 11.1%.

87.9% of the participants are experienced software engineers with more than 5 years experience, as shown in Figure 8. Industrial, open source, and personal projects are the dominant project types that the participants are involved in, followed by academic projects (see Figure 9). Note that a participant can work on more than one type of project.

5.4 Data Analysis

The responses of the survey are available in the online appendix¹⁵. There are 12 open-ended questions in the survey where participants can choose to input their own answers in free text. For each of the question, we use an Open Coding approach to let the coding schema emerge during the analysis (Glaser, 2017). We adopt a three phase coding process:

- Phase I: two of the authors coded the answers of each open-ended question individually. As a result, each of the two authors had his own set of codes for the answers. Then, these two authors discussed their draft code schema and made a revised version of the code schema.

¹⁵ <http://sel.ist.osaka-u.ac.jp/people/wuyuhao/research-data/so-code-reuse/survey-responses.csv>

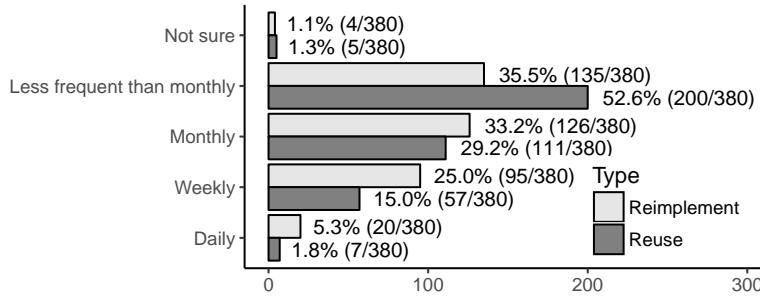


Fig. 10: Comparison of frequency of reusing and reimplementing source code.

- Phase II: the two authors used the revised schema to code the answers. Then, they discussed and resolved conflicts. As a result, a unified coding schema is developed and applied to all the answers.
- Phase III: the third author verified the developed schema and coding results.

5.5 Results

We answer RQ3 and RQ4 in the remainder of this section and RQ5 in Section 6.

5.5.1 RQ3: Do developers prefer reusing or reimplementing the source code, why?

Developers reimplement source code slightly more frequently than reuse source code. Figure 10 shows the comparison of frequency of reimplementing source code and reusing source code from Q&A platforms. The number of participants who reimplement source code monthly (33.2%) and those who reuse source code monthly (29.2%) are close, while the difference increases to 25.0% vs. 15.0% at a weekly frequency.

The majority of developers (65%) prefer reimplementing the source code due to the code modification that is required to make the code from the post work in their project. Table 3 shows the reasons for choosing reimplementing over reusing source code. The top reason that makes developers reimplement source code is the code modification that is required to make the code from the post work in their own projects. This finding is consistent with our finding in RQ 1 (i.e., most code needs modification before reusing) and also confirms the need of research in code integration. Code comprehension ranks as the second most important reason that prevents code reuse. Prior studies also reported that developers spend 58% of their time on program comprehension activities (Xin *et al.*, 2017). Hence, more effort is required to improving code comprehension techniques to further facilitate code

Table 3: Reasons for choosing reimplementing over reusing source code. (Multi-selection allowed, hence the sum of the percentages is larger than 100%.)

Category	Description	Perc.
Context	The code should be written according to its context.	65%
Comprehension	Do not understand the source code to be reused.	44%
Quality	The quality of the source code is too low.	32%
Time consuming	Reusing source code takes more time.	17%
Other	Other reasons.	7%

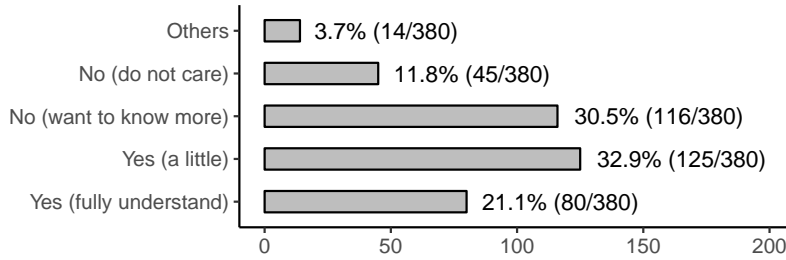


Fig. 11: Participants' awareness of the licenses of Q&A platforms.

reuse from Q&A platforms. 32% of the participants complain about the low code quality on Stack Overflow, which highlights the need for next-generation Q&A platforms to improve the code quality. 17% of the participants mention that reusing source code takes more time than reimplementing it, which is against the common wisdom. One possible reason is that if the source code is big or complex, it would take more time to comprehend it or to make it work in a particular context.

Developers reimplement source code slightly more frequently than reuse source code primarily due to the code context, the difficulty of code comprehension, and the low quality of source code. Our observations provide empirical evidence to support current research on code integration and code comprehension, and highlight the need of improving code quality for next-generation Q&A platforms.

5.5.2 RQ4: What is the influence of the code license of Stack Overflow?

75.3% of the participants do not have a good understanding of the license terms of the Q&A platforms, which indicates that there are potential license violation issues when developers reuse source code from Q&A platforms. Figure 11 shows the results of participants' awareness of the licenses of Q&A platforms. An *et al.* (2017) investigated the code reuse on Android apps and observed 1,279 potential license violation cases where developers reused source code from or to Q&A platforms. Our survey results give a possible explanation for these violations. The "Other" category

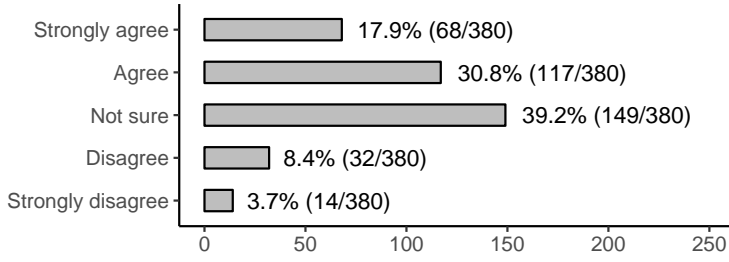


Fig. 12: Participants' opinion of license compatibility between Q&A platforms and their projects.

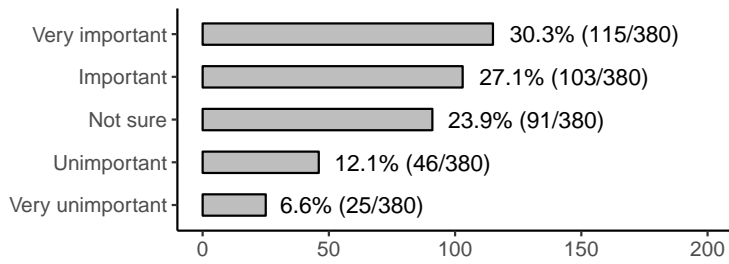


Fig. 13: Importance of having more information on license.

in Figure 11 includes cases in which the participants did not give a concrete answer, e.g., *“Depends on the platform. Stack Overflow is attribution-required, but the requirements of most other sites are vague or not generally known.”*

51.3% of the participants disagree on or are not sure whether the license of Q&A platforms is compatible with their own projects, which indicates that such participants might be prevented from reusing code from Q&A platforms (see Figure 12). We actually observed some participants mentioned this when they were asked why they prefer reimplementing over reusing source code. For example, one participant mentioned that *“licensing is sometimes an issue.”*

Most of the participants (57.4%) think that having more information about the code license is essential (see Figure 13). This echoes with the result that most participants do not have a good understanding of the code license of Q&A platforms, and reveals the needs from the participants that Q&A platforms should make their code license clearer.

Table 4: The categorization of code quality suggestions — 64 (37.6%).

Category	Description (D) – Example (E)	Perc.
Integrated validator	D: Integrated validator that can test the code snippets on Q&A platforms. E: <i>“An inbuilt REPL environment for as many languages/environments as possible.”</i>	42.2%
Outdated code	D: Answers (including source code) on Q&A platforms suffer from out-of-date problems. Participants are seeking for a solution to this problem. E: <i>“make date important in marking outdated code, and deprecate those snippets via the community”</i>	29.7%
Answer quality	D: Classifier that helps distinguish high and low quality answers. E: <i>“Better support for answers that are good, but out of date.”</i>	17.2%
Code review	D: Integrated code review tool that helps improve the code quality. E: <i>“In-browser code review and commenting similar to that provided by commercial code review tools.”</i>	10.9%
Total		100.0%

Generally speaking, participants do not have a good understanding of the code license on Q&A platforms. More than half of them believe that or are not sure of the license compatibility issue exists between their projects and Q&A platforms. Most of the participants think that they need more information of the code license on Q&A platforms. Based on the observations, the next-generation Q&A platforms should have clearer license and make it more visible to developers.

6 A Roadmap for Next-Generation Q&A platforms

In this section, we summarize the results for RQ5. In total, we collected 150 responses for Q19 in the survey. 22 of these responses were not actually suggestions for next-generation Q&A platforms (e.g., *“Not much, quite happy with Stack Overflow.”*) and were excluded from the following analysis. Each response can contain multiple suggestions and we extracted 170 suggestions from these responses. Using the approach described in Section 5.4, the suggestions were then categorized into five major categories: code quality, information enhancement & management, data organization, license, and human factors. We highlight the findings and discuss implications on future research of next-generation Q&A platforms of each category.

6.1 Code Quality

Code quality is the most popular type of suggestion (37.6%) from the participants. From Table 4, we observe that 37.6% of the participants

provide suggestions on improving the code quality on Q&A platforms. The two most important suggestions from developers on improving code quality are adding an integrated validator (42.2%) that can test source code online and an outdated code detection mechanism (29.7%) that can identify outdated code.

An integrated validator is a convenient way of testing source code snippets online to ensure the quality of the source code. Participants describe such tool for example as follows: *“The ability to interact with and run the code examples written in answers and questions”*.

Some Q&A platforms have started to integrate an online validator into their websites. For example, Stack Overflow supports three web-creating languages for online validation: HTML, CSS, and JavaScript (Stack Overflow, 2014). However, Stack Overflow does not support online validator of other languages, such as Java and C++, which are also very popular on Stack Overflow. There are several challenges to extend the validator to all languages.

One of the biggest challenges is to make an incomplete code snippet run correctly, since the code snippets on Q&A platforms are usually not complete as the answerer only needs to implement the core part of a solution and may leave out necessary context information (e.g., version). Prior studies have proposed several approaches to extend incomplete code snippets (not limited to those on Q&A platforms) into compilable ones based on program analysis and machine learning techniques (Wang *et al.*, 2016a; Nguyen *et al.*, 2012; Raychev *et al.*, 2014). However, none of these approaches can guarantee the correctness of the extended code. For example, there is a function called “foo” in a code snippet. To make this code snippet compilable, Q&A platforms need to infer where this function comes from and then import the corresponding library. However, prior approaches cannot automatically infer the exact library only based on the source code. This problem may be solvable in Q&A platforms by leveraging the description that comes with the source code. Hence, future research should investigate whether the description of the source code can be used to improve the correctness of the code extension.

Providing an outdated code detection mechanism is the second top suggestions under the code quality category from the participants. Many of the participants mention that the source code on Q&A platforms is often outdated and not suitable for current technologies or situations. For example, one participant suggests: *“Have explicit mechanisms for dealing with content that goes out of date due to platform or language changes.”* Some participants suggest a mechanism that clarifies the API version of the source code: *“Clear associates between the code snippets the versions of the API under which it will work. This is particularly when working with APIs that change frequently, like iOS and Unity.”* Some also suggest deprecating the outdated answers: *“Make date important in marking outdated code, and deprecate those snippets via the community.”* This problem was recognized by various developers on Stack Overflow (Krumia, 2014) and received wide attention from the Stack Overflow communities.

Table 5: The categorization of Information enhancement & management suggestions — 43 (25.3%).

Category	Description (D) – Example (E)	Perc.
Answer tagging	D: Better tagging-like information system for answers. E: <i>“Provide/require tagging of the version number(s) of the language [...]”</i>	37.2%
Code evolution	D: Better management of the evolution/revisions of code snippets. E: <i>“where does the code come from and copied to, and also the revisions inside the platform.”</i>	14.0%
Resources linking	D: Q&A platforms should suggest for other resources (e.g., books, API documents, libraries etc.) E: <i>“Books suggestions based on questions.”</i>	11.6%
Answer writing support	D: Support for writing better questions/answers. E: <i>“[...] it would be nice if it would be easier to ask a good question [...]”</i>	9.3%
Others	D: Other aspects of information enhancement & management. E: <i>“Built in support within an IDE to make it faster to get the answer you are interested in.”</i>	27.9%
Total		100.0%

However, as we know, no existing study investigates the outdated source code or solutions on Q&A platforms so far. Hence, there is a need for future research to put more effort on developing mechanisms to deal with the outdated source code or solutions. There are two primary directions to deal with the outdated code or solutions. First, future studies should propose approaches to identify the outdated solution in Q&A platforms using machine learning techniques automatically. Second, future research should develop certain incentive systems to motivate communities to identify and update such outdated source code or solutions.

6.2 Information Enhancement & Management

Information enhancement & management (25.3%) is the second most popular suggestion for developers on Q&A platforms. 37.2% of the participants suggest to add tagging-like information for answers (see Table 5). These participants suggest a mechanism (e.g., adding tagging-like information) to identify special answers such as those using a specific version of an API or those which pass a specific test. For example, participants suggest that *“Provide/require tagging of the version number(s) of the language or other context that the questions and answers are written for.”* and *“Answers which pass the tests could be marked with a special icon and given priority.”*

Based on the observations, future studies should focus on recommending the tagging-like information for answers. These recommendations could be

Table 6: The categorization of data organization suggestions — 21 (12.4%).

Category	Description (D) – Example (E)	Perc.
Code searching/indexing	D: Support for easier code search. E: “ <i>Source Code indexing for easier retrieval. It could also give the possibility to find example of usage functions.</i> ”	47.6%
Duplicate posts	D: An automatic way of clustering duplicate questions/answers. E: “ <i>Auto-suggest similar questions, particularly for questions that don’t have answers.</i> ”	38.1%
Comments	D: Support on utilizing the comments of posts. E: “ <i>Code in *comments* must be expressed better, than on Stack Overflow.</i> ”	14.3%
Total		100.0%

made automatically using, e.g., machine learning techniques (Wang *et al.*, 2017a, 2014b; Zhou *et al.*, 2017), or aspect-mining techniques (Wong *et al.*, 2008; Wang *et al.*, 2010; Yu *et al.*, 2011; Liu *et al.*, 2015; Zhao and Li, 2009).

6.3 Data Organization

12.4% of the participants suggest Q&A platforms to better organize their data, including a better searching/indexing mechanism and better duplicate detection. As shown in Table 6, some participants (47.6%) suggest that Q&A platforms should have a better way to index and search code. For example, participants mentioned: “*Source Code indexing for easier retrieval. It could also give the possibility to find example of usage functions.*” and “*Ability to search questions based on the version of the framework or language I’m working with*”. 38.1% of the participants suggest that Q&A platforms should have an automatic way to detect duplicate or similar post and be organized in a better way. Others (14.3%) suggest to have better support on utilizing the comments in posts.

Prior studies have studied develop code search engines to help developers to improve their search efficiency on source code (Wang *et al.*, 2014a; McMillan *et al.*, 2011; Lv *et al.*, 2015; Bajracharya *et al.*, 2006; Searchcode, 2016b). Our findings support these studies, and it would be interesting to integrate these code search engines into Q&A platforms.

Prior studies proposed various approaches to help Q&A platforms detect duplicate questions automatically (Zhang *et al.*, 2015; Ahasanuzzaman *et al.*, 2016; Wang *et al.*, 2017b; Zhang *et al.*, 2017). Our findings support these studies. Moreover, deep learning has proven its power on capturing the semantic meaning from natural language in many prior studies (Ganguly *et al.*, 2015; Lai *et al.*, 2015; Bian *et al.*, 2014; Chen *et al.*, 2016). The common way to identify the duplicate questions is to measure such questions’ similarity in terms

Table 7: The categorization of code license suggestions — 23 (13.5%).

Category	Description (D) – Example (E)	Perc.
Clearer license	D: Q&A platforms should make their license terms clearer. E: <i>“By far the most important requirement is clear licensing. Much of the code provided on such platforms is not currently usable because the license is unclear.”</i>	69.6%
Permissive license	D: Q&A platforms should use a more permissive license. E: <i>“Let the user choose a more re-user-friendly license (e.g. copy without reference).”</i>	30.4%
Total		100.0%

of semantic meaning. Hence, future research could consider to employ deep learning to detect duplicate questions or find similar questions.

6.4 Code License

13.5% of the participants suggest to improve license-related issues, in particular to make the license more clear (69.6%). Table 7 shows the suggestions about license. This trend echoes with our previous result that 75.3% of the participants do not have a good understanding of the license terms of the Q&A platforms (see Section 5.5.2). Participants request that the Q&A platforms should provide a clear explanation of their license terms: *“By far the most important requirement is clear licensing. Much of the code provided on such platforms is not currently usable because the license is unclear.”* If developers neglect the license of the Q&A platforms and reuse source code from these platforms, they are under the risk of license violation which may cause them problems later.

30.4% of the participants suggest that Q&A platforms should use a more permissive license which has less restrictions on source code reuse. In the example of Stack Overflow, CC BY-SA 3.0 was the original license for the source code on this platform. CC BY-SA 3.0 is a copyleft (non-permissive) license which requires the derivative work be licensed under the same license (CC BY-SA 3.0). This means that when developers reuse the source code from Stack Overflow into their projects, they have to license these projects under CC BY-SA 3.0 as well. Otherwise, they are under the risk of license violation.

It is also worth noting that, although Stack Overflow has announced this license change in the post on Stack Exchange (Stack Exchange, 2015), the community did not reflect this change on their homepage (Stack Overflow, 2017), where it still writes *“user contributions licensed under cc by-sa 3.0 with attribution required”*. This mismatch will further deepen developers’ misunderstanding of the license terms, hence we suggest that next-generation Q&A platforms should explicitly describe their license terms of source code reuse in a consistent manner.

Table 8: The categorization of human factor suggestions — 19 (11.2%).

Category	Description (D) – Example (E)	Perc.
Better curator	D: Better curators are needed to help improve the quality of the posts. E: <i>“Definitely curators for specific languages to rate answers in specific areas.”</i>	63.2%
Gamification related	D: Suggestions on improving the gamification system of the Q&A platforms. E: <i>“Base reputation on number of answers up-voted by others, not on personal activity.”</i>	36.8%
Total		100.0%

6.5 Human factor

11.2% of the participants suggest to improve Q&A platforms in terms of Human factor. Table 8 shows that 63.2% of the participants suggest to have better curators to improve the quality of posts and help with marking good answers. One participant suggested: *“Pay some vetted, experienced developers to check the answers, instead of relying on gamification.”* Another suggestion also emphasized on the collaboration from the community: *“Arriving at a ‘most correct’ solution should be a more collaborative effort with a clearly shown path of how it was arrived at by multiple people, not necessarily just one user who takes all the credit.”*

36.8% of the participants suggest to improve the gamification system of Q&A platforms. The usage of gamification on Q&A platforms has been proven effective before (Anderson *et al.*, 2013; Cavusoglu *et al.*, 2015). However, participants revealed some flaws in this system. For example, one participant wrote: *“Base reputation on number of answers up-voted by others, not on personal activity. (Stack Overflow has too many nit-pickers gaining reputation by down-voting legitimate questions.)”* This suggestion echoes with another participant’s comments: *“Gamification sometimes creates wrong incentives.”*

Our findings suggest that future studies are necessary on how to improve the gamification mechanism of Q&A platforms.

7 Related Work

In this section we discuss prior research that are related to our study.

7.1 Leveraging the Knowledge from Q&A Platforms

Q&A platforms accumulate a large amount of knowledge from the communication among developers, which can be used to assist software development activities. Thus, numerous studies about how to leverage the knowledge from Q&A platforms have been done. Treude and Robillard (2016) presented a

novel machine learning based approach, named SISE, to augment API documentation using the answers on Stack Overflow. Wong *et al.* (2013) leveraged questions and answers on Stack Overflow to automatically generate comments in system source code. Vassallo *et al.* (2014) extracted discussions from Stack Overflow and used the extracted data to generate JavaDoc. Gao *et al.* (2015) proposed an automated approach to fix recurring crash bugs by leveraging information (e.g., questions with similar crash traces) on Q&A platforms. Azad *et al.* (2017) proposed an approach to extract API call rules from version history and Stack Overflow Posts. Chen *et al.* (2017) proposed an automatic approach to build a thesaurus that contains morphological forms of software engineer terms.

Several studied the attached source code on Stack Overflow. Sillito *et al.* (2012) performed an empirical study on factors that make a good source code example on Stack Overflow. They found that the source code explaining important elements and presenting a solution step by step makes a good example. Treude *et al.* (2011) performed a study on Stack Overflow to explore which questions are answered well and which ones remain unanswered. They found that the code is an important factor for “code review” questions to get an good answer. Abdalkareem *et al.* (2017) performed a study on Stack Overflow to explore what developers use the crowd for. They found that development tools and programming language issues are areas where the crowd is the most helpful.

Different from above studies which focus on developing a tool to deliver knowledge from Q&A platforms to developers, our study focuses on understanding how developers utilize the source code from Q&A platforms.

7.2 Source Code Reuse

Rigby and Robillard (2013) proposed a novel approach to extract code elements from various documents such as Stack Overflow. They evaluated their approach on 188 Stack Overflow answer posts containing 993 code elements. The technique achieved an average 0.92 precision and 0.90 recall. Ponzanelli *et al.* (2013) proposed an Eclipse plugin named SEAHAWK that helps developers search and import code snippets from Stack Overflow. Then they proposed a Eclipse plugin named Prompter which automatically searches and identifies Stack Overflow discussions, evaluates their relevance based on the given code context in the IDE, and notifies the developer if a user-defined confidence threshold is surpassed (Ponzanelli *et al.*, 2014a,b). Armaly and McMullan (2016) presented a novel reuse technique that allows programmers to reuse functions from a C or C++ program, by recording the state of the dependencies during one program’s execution, and replaying them in the context of a different program.

Different from prior work which focus on proposing approaches to retrieve code for developers to reuse, we study how developers utilize the source code

from Q&A platforms. We also performed a user survey to understand what factors prevent developers to reuse the source code from Q&A platforms.

7.3 Code Licensing

Baltes *et al.* (2017) studied whether developers attribute (e.g., reference) the original Stack Overflow post when they reuse the code snippets from these posts. They found that 3.22% of all the analyzed repositories and 7.33% of the popular ones on GitHub contained a reference to Stack Overflow. However, for Java, at least one third of the copied snippets were not attributed. An *et al.* (2017) studied whether developers respect license restrictions when reusing source code from Stack Overflow in Android apps or vice versa. With a case study on 399 Android apps, they found 232 code snippets in 62 Android apps were potentially reused from Stack Overflow, while 1,226 Stack Overflow posts contained code in 68 Android apps. 1,279 potential license violations were observed. Almeida *et al.* (2017) performed a survey among developers on the open source license and found that developers struggle when multiple licenses were involved. The results indicate a need for tool support to help guide developers in understanding this critical information of license that is attached to software components.

Different from previous studies on code license, we focus on studying whether the license has impact on source code reuse from Q&A platforms for developers and we find it is a potentially important factor that prevents code reuse from Q&A platforms.

8 Threats to Validity

External validity. Threats to external validity relates to the generalizability of our findings. In this research, we use `searchcode.com` as our search engine and find 1,059 source files in total that contain links to Stack Overflow posts. After removing the small projects and duplicate files, there are 95 JavaScript files left. The number of files may not be large enough to represent all the cases in the real world. However, we think the number is large enough for our exploratory study to find out the issues underlying the process of code reuse from Q&A platforms.

We focused the first part of our study on JavaScript language, and our result may not be generalizable to other languages. Future work is necessary to investigate whether our findings hold for other languages.

Internal validity. Threats to internal validity relates to the experimenter bias and errors. In this research, we manually inspect the source code in the projects and in Stack Overflow posts. We also manually categorize each case of how developers are reusing the source code. Different people may have different opinions on the categorization. To alleviate this threat, two of our authors conduct the manual analysis and any discrepancy is discussed till

a consensus is achieved. When analysing the results of user survey, we also manually categorized the open questions in survey. To alleviate the bias due to human factors, we employ the open coding process.

9 Conclusion

Prior studies focus on developing approaches to help developers reuse and integrate source code in a required context automatically. However, little to nothing is known about how developers utilize the code from Q&A platforms.

In this paper, we study how developers reuse code from Q&A platforms and the barriers that prevent developers from reusing source code from Q&A platforms.

We first conduct an exploratory study on 95 JavaScript files from 89 projects which contain source code that contains an explicit reference to a Stack Overflow post with the expectation of finding how developers utilize Q&A platforms. Based on the result of this exploratory study, we conduct a survey on 453 developers on Stack Overflow to further understand the barriers that may prevent code reuse and to obtain suggestions for the next-generation Q&A platforms. The most important findings of our study are:

1. The exploratory study shows that 78.2% of the reused source code from Stack Overflow is modified from simple refactoring to a complete reimplementation.
2. 26% of the developers also reuse source code from non-accepted answers.
3. More developers prefer reimplementing source code over reusing source code because of different code context, low code quality and difficulty of code comprehension.
4. Code quality, information enhancement & management, data organization, code license and human factor are the most popular types of suggestions for next-generation Q&A platforms from the survey participants.

Future research efforts can be devoted to a next-generation Q&A platform based on the suggestions summarized in this work.

References

- Abdalkareem, R., Shihab, E., and Rilling, J. (2017). What do developers use the crowd for? a study using Stack Overflow. *IEEE Software*, **34**(2), 53–60.
- Ahasanuzzaman, M., Asaduzzaman, M., Roy, C. K., and Schneider, K. A. (2016). Mining duplicate questions in Stack Overflow. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR)*, pages 402–412.
- Almeida, D. A., Murphy, G. C., Wilson, G., and Hoyer, M. (2017). Do software developers understand open source licenses? In *Proceedings of the 25th International Conference on Program Comprehension (ICPC)*, pages 1–11. IEEE.

- Alnusair, A., Rawashdeh, M., Hossain, M. A., and Alhamid, M. F. (2016). Utilizing semantic techniques for automatic code reuse in software repositories. In *Quality Software Through Reuse and Integration*, pages 42–62. Springer.
- An, L., Mlouki, O., Khomh, F., and Antoniol, G. (2017). Stack Overflow: A code laundering platform? In *Proceedings of the 24th IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 283–293. IEEE.
- Anderson, A., Huttenlocher, D., Kleinberg, J., and Leskovec, J. (2013). Steering user behavior with badges. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pages 95–106. ACM.
- Armaly, A. and McMillan, C. (2016). Pragmatic source code reuse via execution record and replay. *Journal of Software: Evolution and Process*, **28**(8), 642–664.
- Atwood, J. (2009). Attribution required – Stack Overflow blog. <https://stackoverflow.blog/2009/06/25/attribution-required/>. (last visited: Aug 25, 2017).
- Azad, S., Rigby, P. C., and Guerrouj, L. (2017). Generating API call rules from version history and stack overflow posts. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, **25**(4), 29.
- Bajracharya, S., Ngo, T., Linstead, E., Dou, Y., Rigor, P., Baldi, P., and Lopes, C. (2006). Sourcerer: A search engine for open source code supporting structure-based search. In *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 681–682. ACM.
- Baltes, S., Kiefer, R., and Diehl, S. (2017). Attribution required: Stack Overflow code snippets in GitHub projects. In *Proceedings of the 39th International Conference on Software Engineering Companion*, pages 161–163. IEEE.
- Bian, J., Gao, B., and Liu, T.-Y. (2014). *Knowledge-Powered Deep Learning for Word Embedding*, pages 132–148. Springer Berlin Heidelberg.
- Cavusoglu, H., Li, Z., and Huang, K.-W. (2015). Can gamification motivate voluntary contributions?: The case of StackOverflow Q&A community. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*, pages 171–174. ACM.
- Chen, C., Gao, S., and Xing, Z. (2016). Mining analogical libraries in Q&A discussions - incorporating relational and categorical knowledge into word embedding. In *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 338–348. IEEE.
- Chen, C., Xing, Z., and Wang, X. (2017). Unsupervised software-specific morphological forms inference from informal discussions. In *Proceedings of the 39th International Conference on Software Engineering (ICSE)*, pages 450–461. IEEE.
- Cottrell, R., Walker, R. J., and Denzinger, J. (2008). Semi-automating small-scale source code reuse via structural correspondence. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT)*, pages 214–225. ACM.

- Feldthaus, A. and Møller, A. (2013). Semi-automatic rename refactoring for javascript. In *Proceedings of the 2013 ACM SIGPLAN International Conference On Object Oriented Programming Systems Languages & Applications*, volume 48, pages 323–338. ACM.
- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Ganguly, D., Roy, D., Mitra, M., and Jones, G. J. (2015). Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 795–798.
- Gao, Q., Zhang, H., Wang, J., Xiong, Y., Zhang, L., and Mei, H. (2015). Fixing recurring crash bugs via analyzing Q&A sites. In *Proceedings of the 30th International Conference on Automated Software Engineering (ASE)*, pages 307–318.
- Glaser, B. (2017). *Discovery of grounded theory: Strategies for qualitative research*. Routledge.
- Hua, L., Kim, M., and McKinley, K. S. (2015). Does automated refactoring obviate systematic editing? In *IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*, volume 1, pages 392–402. IEEE.
- Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., and Damian, D. (2014). The promises and perils of mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*, pages 92–101. ACM.
- Krumia (2014). Introduce an “obsolete answer” vote. <https://meta.stackoverflow.com/questions/272651/introduce-an-obsolete-answer-vote>. (last visited: Aug 25, 2017).
- Lai, S., Xu, L., Liu, K., and Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2267–2273. AAAI Press.
- Liu, P., Joty, S. R., and Meng, H. M. (2015). Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1433–1443. The Association for Computational Linguistics.
- Lv, F., Zhang, H., Lou, J.-g., Wang, S., Zhang, D., and Zhao, J. (2015). CodeHow: Effective code search based on API understanding and extended boolean model. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 260–270. IEEE.
- McMillan, C., Grechanik, M., Poshyvanyk, D., Xie, Q., and Fu, C. (2011). Portfolio: Finding relevant functions and their usage. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, pages 111–120.
- Meng, N., Kim, M., and McKinley, K. S. (2011). Systematic editing: Generating program transformations from an example. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and*

- Implementation (PLDI)*, pages 329–342.
- Meng, N., Kim, M., and McKinley, K. S. (2013). Lase: locating and applying systematic edits by learning from examples. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 502–511. IEEE.
- Nguyen, A. T., Nguyen, T. T., Nguyen, H. A., Tamrawi, A., Nguyen, H. V., Al-Kofahi, J., and Nguyen, T. N. (2012). Graph-based pattern-oriented, context-sensitive source code completion. In *Proceedings of the 34th International Conference on Software Engineering (ICSE)*, pages 69–79.
- Ponzanelli, L., Bacchelli, A., and Lanza, M. (2013). Leveraging crowd knowledge for software comprehension and development. In *Proceedings of the 17th European Conference on Software Maintenance and Reengineering (CSMR)*, pages 57–66. IEEE.
- Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., and Lanza, M. (2014a). Mining stackoverflow to turn the IDE into a self-confident programming prompter. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 102–111. ACM.
- Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., and Lanza, M. (2014b). Prompter: A self-confident recommender system. In *ICSME*, pages 577–580.
- Raychev, V., Vechev, M., and Yahav, E. (2014). Code completion with statistical language models. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 419–428.
- Rigby, P. C. and Robillard, M. P. (2013). Discovering essential code elements in informal documentation. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE)*, pages 832–841. IEEE.
- Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering (TSE)*, **25**(4), 557–572.
- Seaman, C. B., Shull, F., Regardie, M., Elbert, D., Feldmann, R. L., Guo, Y., and Godfrey, S. (2008). Defect categorization: making use of a decade of widely varying historical data. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 149–157. ACM.
- Searchcode (2016a). searchcode - API. <https://searchcode.com/api/>. (last visited: Aug 25, 2017).
- Searchcode (2016b). searchcode - Homepage. <https://searchcode.com/>. (last visited: Aug 25, 2017).
- Sillito, J., Maurer, F., Nasehi, S. M., and Burns, C. (2012). What makes a good code example?: A study of programming Q&A in StackOverflow. In *Proceedings of the 2012 IEEE International Conference on Software Maintenance (ICSM)*, pages 25–34.
- Stack Exchange (2015). The MIT license — clarity on using code on Stack Overflow and Stack Exchange. <https://meta.stackexchange.com/q/271080/337948>. (last visited: Aug 25, 2017).
- Stack Exchange (2017). All sites - Stack Exchange. <https://stackexchange.com/sites>. (last visited: Aug 25, 2017).

- Stack Overflow (2014). Feedback requested: Runnable code snippets in questions and answers. <https://meta.stackoverflow.com/questions/269753/feedback-requested-runnable-code-snippets-in-questions-and-answers>. (last visited: Aug 25, 2017).
- Stack Overflow (2016). Stack Overflow developer survey results 2016. <http://stackoverflow.com/research/developer-survey-2016>. (last visited: Aug 25, 2017).
- Stack Overflow (2017). Stack Overflow - Homepage. <https://stackoverflow.com/>. (last visited: Aug 25, 2017).
- Treude, C. and Robillard, M. P. (2016). Augmenting API documentation with insights from Stack Overflow. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, pages 392–403. ACM.
- Treude, C., Barzilay, O., and Storey, M.-A. (2011). How do programmers ask and answer questions on the web? (NIER track). In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, pages 804–807.
- Vasilescu, B., Filkov, V., and Serebrenik, A. (2013). StackOverflow and GitHub: Associations between software development and crowdsourced knowledge. In *Proceedings of 2013 International Conference on Social Computing (SocialCom)*, pages 188–195. IEEE.
- Vassallo, C., Panichella, S., Di Penta, M., and Canfora, G. (2014). CODES: Mining source code descriptions from developers discussions. In *Proceedings of the 22nd International Conference on Program Comprehension (ICPC)*, pages 106–109.
- Wang, H., Lu, Y., and Zhai, C. (2010). Latent aspect rating analysis on review text data: A rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 783–792.
- Wang, S., Lo, D., and Jiang, L. (2014a). Active code search: Incorporating user feedback to improve code search relevance. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE)*, pages 677–682.
- Wang, S., Lo, D., Vasilescu, B., and Serebrenik, A. (2014b). EnTagRec: An enhanced tag recommendation system for software information sites. In *Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 291–300.
- Wang, S., Lo, D., and Jiang, L. (2016a). Autoquery: automatic construction of dependency queries for code search. *Automated Software Engineering*, **23**(3), 393–425.
- Wang, S., Lo, D., Vasilescu, B., and Serebrenik, A. (2017a). EnTagRec ++: An enhanced tag recommendation system for software information sites. *Empirical Software Engineering*.
- Wang, Y., Feng, Y., Martins, R., Kaushik, A., Dillig, I., and Reiss, S. P. (2016b). Hunter: next-generation code reuse for java. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 1028–1032. ACM.

- Wang, Z., Hamza, W., and Florian, R. (2017b). Bilateral multi-perspective matching for natural language sentences. *CoRR*, **abs/1702.03814**.
- Wong, E., Yang, J., and Tan, L. (2013). Autocomment: Mining question and answer sites for automatic comment generation. In *IEEE/ACM 28th International Conference on Automated Software Engineering (ASE)*, pages 562–567. IEEE.
- Wong, T.-L., Lam, W., and Wong, T.-S. (2008). An unsupervised framework for extracting and normalizing product attributes from multiple web sites. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 35–42.
- Xin, X., Lingfeng, B., David, L., Zhenchang, X., Ahmed, E. H., and Shanping, L. (2017). Measuring program comprehension: A large-scale field study with professionals. *IEEE Transactions on Software Engineering (TSE)*, **99**(26).
- Yellin, D. M. and Strom, R. E. (1997). Protocol specifications and component adaptors. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, **19**(2), 292–333.
- Yu, J., Zha, Z.-J., Wang, M., and Chua, T.-S. (2011). Aspect ranking: Identifying important product aspects from online consumer reviews. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 1496–1505.
- Zhang, W. E., Sheng, Q. Z., Lau, J. H., and Abebe, E. (2017). Detecting duplicate posts in programming qa communities via latent semantics and association rules. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, pages 1221–1229.
- Zhang, Y., Lo, D., Xia, X., and Sun, J.-L. (2015). Multi-factor duplicate question detection in Stack Overflow. *Journal of Computer Science and Technology*, **30**(5), 981–997.
- Zhao, L. and Li, C. (2009). *Ontology Based Opinion Mining for Movie Reviews*, pages 204–214. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Zhou, P., Liu, J., Yang, Z., and Zhou, G. (2017). Scalable tag recommendation for software information sites. In *Proceedings of the 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 272–282. IEEE.

Appendix

Below are the questions and options in our online survey. Single-selection options are marked with circle marks (○) in front; multi-selection options are marked with box marks (□) in front. When participants choose the option “Other”, they are allowed to input a free text as an additional answer.

Part I

1. How many years of software engineering experience do you have?

○ 1	○ 2	○ 3	○ 4	○ 5
○ 6	○ 7	○ 8	○ 9	○ 10+
2. What type of project(s) are you working on?

- Open source Personal Industrial Academic Other
3. Which programming language(s) do you use in your projects?
- Java C/C++ C# Python Other
4. How often do you use Q&A platforms?
- Every day Once a week Once a month
 Once every few months or less Never
5. What do you use Q&A platforms for?
- Learning new techniques/methodologies
 Refreshing the knowledge of old techniques/methodologies
 Solving a specific programming issue
 Finding references that I can refer to in my source code to make future maintenance easier
 Answering questions
 Other
6. Which Q&A platforms do you use to look for solutions to programming-related issues?
- Stack Overflow
 Quora
 Product-specific support forums
 Language-specific support forums
 I do not use Q&A platforms for this purpose
 Other
7. Have you ever reused source code from a Q&A platform?
- Yes No

Part II

8. How often do you *reuse* source code from Q&A platforms?
- Every day Once a week Once a month
 Once every few months or less Never
9. How often do you *reimplement* source code from Q&A platforms?
- Every day Once a week Once a month
 Once every few months or less Never
10. If you prefer reimplementing the source code over reusing existing code, why?
- Code should be written in relation to the context.
 I don't want to reuse source code that I don't fully comprehend.
 The quality of the existing source code is too low.
 Re-implementing the source code takes less time than reusing.
 Other
11. What do you consider the most important factors when deciding when to reuse code from a Q&A platform?
- Correctness (i.e., bug-free)
 Performance (i.e., efficient)
 Readability (i.e., easy to read/understand)
 Simplicity (i.e., less lines of code)
 Compatibility (e.g, support more platforms)
 Whether the answer is accepted by the questioner.
 Whether the answer has the highest number of upvotes.
 Other
12. Which aspects cause you difficulty when reusing source code from Q&A platforms?
- Syntax errors need to be fixed to make the source code runnable.
 Bugs (e.g., index out of bounds) need to be fixed.
 Readability needs to be improved.
 Performance needs to be improved.
 The code snippet is not in the programming language I need.
 Code needs to be adapted to my specific use case.
 The license terms of the Q&A platform are unclear.
 Other
13. Do you always refer to the Q&A platform post from which you reused source code in your documentation or code comments? Why (not)?

- Yes, I add a link to the post/answer to show my respects/appreciation to the original author.
 - Yes, because it is required by the license terms of that Q&A platform (e.g., CC-BY-SA 3.0 in the case of Stack Overflow).
 - Yes, to make it easier for future maintenance.
 - No, I would like to, but always forget.
 - No, I don't do that. (Please elaborate the reason below if you could)
 - Other
14. Are you aware of the license terms of reused source code from a Q&A platform?
- Yes, I fully understand the license terms.
 - Yes, I know about the existence of such terms, but I am not sure what obligations I have.
 - No, I did not know about them, but I would like to learn more about them.
 - No, I did not know about them, neither do I care about them.
 - Other
15. Which license(s) do the projects into which you reused code from a Q&A platform use?
- GPL family (any version of LGPL, GPL or AGPL) MIT License
- Apache License BSD License
- A proprietary license No license
- I don't know Other
16. In general, would you say that the license(s) of these project(s) are compatible with the license of the Q&A platform from which you reused source code?
- Strongly disagree ○ Disagree ○ Neutral
 - Strongly agree ○ Agree
17. How important is it to have more detailed information about the license terms and legal obligations of reusing source code from Q&A platforms?
- Very unimportant ○ Unimportant ○ Neutral
 - Very important ○ Important

Part III

18. How useful would it be to let other users tag answers on Q&A platforms with labels that describe the source code in an answer as 'performant', 'correct', 'readable', etc.?
- Very unuseful ○ Unuseful ○ Neutral
 - Very useful ○ Useful
19. If you could give any suggestions (regardless of whether they are feasible) for a next-generation code Q&A platform, what would you suggest?
-