Noname manuscript No.
(will be inserted by the editor)

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 source code from Q&A platforms, we can provide insights for researchers which can be used to improve next-generation Q&A platforms to help developers reuse source code fast and easily.

In this paper, we first conduct an exploratory study on 289 files from 182 open-source projects, which contain source code that has an explicit reference to a Stack Overflow post. Our goal is to understand how developers utilize code from Q&A platforms and to reveal barriers that may make code reuse more difficult. In 31.5% of the studied files, developers needed to modify source code from Stack Overflow to make it work in their own projects. The degree of required modification varied from simply renaming variables to rewriting the whole algorithm. Developers sometimes chose to implement an algorithm from scratch based on the descriptions from Stack Overflow answers, even if there was an implementation readily available in the post. In 35.5% of the studied files, developers used Stack Overflow posts as an information source for later reference.

To further understand the barriers of reusing code and to obtain suggestions for improving the code reuse process on Q&A platforms, we conducted a survey with 453 open-source developers who are also on Stack Overflow. We

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

found that the top 3 barriers that make it difficult for developers to reuse code from Stack Overflow are: (1) too much code modification required to fit in their projects, (2) incomprehensive code, and (3) low code quality.

We summarized and analyzed all survey responses and we identified that developers suggest improvements for future Q&A platforms along the following dimensions: code quality, information enhancement & management, data organization, license, and the human factor. For instance, developers suggest to improve the code quality by adding an integrated validator that can test source code online, and an outdated code detection mechanism. Our findings can be used as a roadmap for researchers and developers to improve code reuse.

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). For example, these are two of the challenges that developers face when reusing source code from Q&A platforms:

(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,c).
(2) Even if developers are able to find suitable source code, it may be difficult to integrate the code in their own projects. For example, parameters may need to be adjusted, or additional source code may need to be added (Cottrell *et al.*, 2008). To address the challenge of code integration, various techniques have been proposed (Cottrell *et al.*, 2008; Meng *et al.*, 2011; Yellin and Strom, 1997; Meng *et al.*, 2013; Hua *et al.*, 2015), such as automatically renaming variables to make the code fit in the required context.

Prior studies (Gharehyazie *et al.*, 2017; Treude and Robillard, 2017; Xia *et al.*, 2017; Barzilay, 2011) on reusing source code from Q&A platforms have mostly focused on helping developers to locate relevant source code, and on integrating that source code into their own project. In this study, we build upon these studies by investigating *how* developers utilize code from Q&A platforms. Such knowledge will help us better understand the potential barriers that developers face when reusing code from Q&A platforms. In this paper, we use this knowledge to provide a roadmap for improving source code reuse on next-generation Q&A platforms.

We first conduct an exploratory study of 289 source code files from 182 open-source projects, which contain at least one link to a Stack Overflow post. We manually study each file and its linked Stack Overflow post, to investigate how developers reuse code from Stack Overflow. We found that:

- In 44% of the studied files, source code had to be modified before it could be used in the developer's own project. The required modification varied from simple refactorings to a complete reimplementation. This finding provides empirical evidence for the importance of prior studies on automatic code integration (Feldthaus and Møller, 2013; Alnusair *et al.*, 2016; Wang *et al.*, 2016b).
- In 12.5% of the studied files, developers reimplemented 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 a question and its answers.
- Developers reuse source code from non-accepted answers (26%) for several reasons, such as 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 or performance) of answers or adding tags for answers.

To further understand the barriers that developers face when reusing code and to collect suggestions for improving the code reuse process on Q&A platforms, we conducted a survey of 453 open-source developers who are also on Stack Overflow. We highlight our findings as follows:

- **Slightly more participants prefer reimplementing source code over reusing source code from Q&A platforms. The reasons are the difficulty of having to make the code fit in their own projects, and a low comprehension or low quality of the code from Q&A platforms.** These findings provide empirical evidence for the importance of research on code integration and code comprehension, and highlight the need of providing code quality indicators on next-generation Q&A platforms.
- **80% of the participants do not have a good understanding of the licenses of the Q&A platforms. In addition, 57% of the participants think that having more information about the code license is important.** These findings suggest that next-generation Q&A platforms should make code licensing information more visible to developers.
- **The most popular suggestion category for improving code reuse on next-generation Q&A platforms was code quality (35% of the**

suggestions). We categorized the suggestions for next-generation Q&A platforms into five categories: code quality, information enhancement & management, data organization, license, and human factor. A large part of the code quality suggestions were about adding an online code validator (42.2%) and a detection mechanism for outdated source code (29.7%).

In summary, the difficulty of fitting code in their own projects, a low comprehension and a low quality of the code are the top barriers that prevent developers from reusing code. Lacking a good understanding of code license is also an important barrier for code reuse. Thus, future studies are encouraged to address these barriers to better facilitate code reuse for developers.

The rest of this paper is organized as follows. Section 2 introduces the background and related work of our study. Section 3 introduces our research questions and describes our data collection process. Our exploratory study is described in Section 4. Section 5 presents the survey design. Section 6 summarizes and analyzes the survey results, and presents our roadmap for next-generation Q&A platforms. Section 7 describes the threats to validity. And finally, Section 8 concludes the paper.

2 Background & Related Work

In this section, we give background information and discuss related work about one of the most popular technical Q&A platforms, Stack Overflow. We discuss Stack Overflow along four dimensions: leveraging knowledge, understanding the quality of posts, source code reuse, and code licensing.

2.1 Leveraging Knowledge from 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; Wang *et al.*, 2017c). Developers are allowed to post questions, answer questions and vote on questions or answers on Q&A platforms. When posting questions or answers, developers often attach snippets of source code to explain their questions or answers along with the textual description. For example, Figure 1 shows an example of an answer that contains source code on Stack Overflow. The asker asked how to get the HTML of a selected object with the jQuery library, and the answerer posted an answer that provides a solution in the form of the attached source code.

During the process of asking and answering questions, Stack Overflow accumulates a large amount of knowledge. To leverage the vast amount of knowledge on Stack Overflow, several approaches have been proposed. Treude and Robillard (2016) presented a machine learning based approach, SISE, to augment API documentation using answers on Stack Overflow. Gao *et al.* (2015)

Listing 1: Source code reuse example.

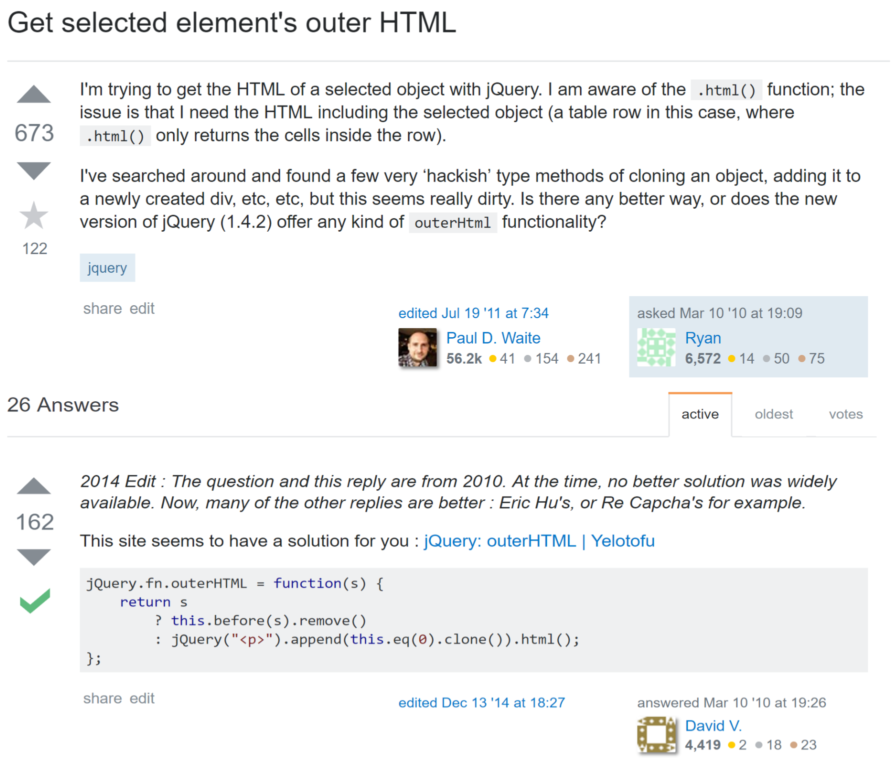
```
//http://stackoverflow.com/questions/2419749/get-
// selected-elements-outer-html
jQuery.fn.outerHTML = function (s) {
  return s
    ? this.before(s).remove()
    : jQuery("<p>").append(this.eq(0).clone()).html();
};
```

proposed an automated approach to fix recurring crash bugs by leveraging information (e.g., questions with similar crash traces) on Stack Overflow. 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 engineering terms. These studies make use of the text and code information on Stack Overflow to automatically generate or enrich existing software artifacts and show promising results. Our paper is different from these studies as we are interested in studying *how* developers utilize knowledge (i.e., source code) from Stack Overflow.

2.2 Understanding the Quality of Posts on Stack Overflow

Stack Overflow allows askers to mark at most one answer as the “accepted answer” to indicate whether the answer meets their requirement (see Figure 1). Stack Overflow allows developers to upvote or downvote a post (e.g., a question or an answer) to express whether the post is useful. The total number of up and downvotes that a post receives is displayed as a score next to the post. For example, the score of the question in Figure 1 is 673. In general, answers and questions with a high score are usually regarded as high-quality ones.

Several studies have been done to investigate the quality of questions and answers on Q&A platforms (Sillito *et al.*, 2012; Treude *et al.*, 2011; Ponzanelli *et al.*, 2014c; Wang *et al.*, 2017b). Sillito *et al.* (2012) performed an empirical study on factors that make a good source code example on Stack Overflow. They found that explaining important elements and presenting a solution step-by-step make 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 source code is an important factor for “code review” questions to get a good answer. Ponzanelli *et al.* (2014c) studied the factors that potentially affect the quality of questions on Stack Overflow. They showed that the attached source code is an important factor for question quality. In this paper we study whether developers tend to reuse source code from a high-quality answer (i.e., a high-voted answer).



Get selected element's outer HTML

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).

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?

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

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();
};
```

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.

2.3 Source Code Reuse from Stack Overflow

Source code reuse can be commonly observed (An *et al.*, 2017). Listing 1 shows an example of code reuse. This source code snippet, which is taken from a GitHub project[1] , is reused from the Stack Overflow post shown in Figure 1.

To help developers reuse source code, several approaches have been proposed. Rigby and Robillard (2013) proposed an approach to extract code elements from various documents such as Stack Overflow posts. They evaluated their approach on 188 Stack Overflow posts containing 993 code elements. Their 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 an Eclipse plugin named Prompter which automatically searches and identifies Stack Overflow discussions, evaluates their relevance based on the given the code context in the IDE, and notifies the developer if a user-defined confidence threshold is surpassed (Ponzanelli *et al.*, 2014a,b). Armaly and McMillan (2016) presented a novel reuse technique that allows programmers to reuse

[1] `https://goo.gl/X84SFi`

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 studies which focused on proposing approaches to retrieve code for developers to reuse, we are interested in studying how developers utilize the source code from Q&A platforms and which barriers they face when doing so.

2.4 Code Licensing on Stack Overflow

Understanding the license of a source code snippet is an important part of code reuse. Developers need to adhere to certain licenses (e.g., MIT and CC BY-SA 3.0) when reusing code from Q&A platforms. For example, 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) license[2], 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 link to the original Stack Overflow post in their source code comments.

A significant number of studies have been performed on code licensing. 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 of 399 Android apps, they found 232 code snippets in 62 Android apps which were potentially reused from Stack Overflow, while 1,226 Stack Overflow posts contained code from 68 Android apps. In total, An et al. observed 1,279 potential license violations. Almeida *et al.* (2017) performed a survey among developers on open-source licenses 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 about the license that is attached to software components. In this study, we would like to understand (1) whether developers are aware about the code license of Stack Overflow and (2) whether this code license forms a barrier for code reuse from Q&A platforms for developers.

3 Research Questions & Data Collection

In this section, we present our research questions and their motivation. In addition, we describe how we collected the datasets that we use to answer our research questions.

[2] `https://creativecommons.org/licenses/by-sa/3.0/`

3.1 Research Questions

We focus on the following five research questions. The first two research questions are answered through an exploratory study on code reuse, in which we collect empirical evidence about code reuse from Stack Overflow in open-source projects. The last three research questions are answered through a survey, in which we contact developers of open-source projects who are active on Stack Overflow as well.

RQ1: To what extent do developers need to modify source code from Stack Overflow in order to make it work in their own projects?
Prior research has proposed several ways to utilize the source code from 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 Eclipse IDE 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 need to 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.

RQ2: From which part of the Stack Overflow post does the reused source code come?
Intuitively, we may expect that accepted answers, or answers with a high score are the most useful. In this RQ, we study whether this intuition is correct. In particular, we investigate why developers chose to reuse code from non-accepted or low-scored answers. By understanding why developers chose non-accepted or low-scored answers, we can provide insights to help Q&A platforms organize their answers better so that developers can find solutions more easily.

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. In this RQ, we survey developers about the correctness of this intuition. We also investigate which factors make developers prefer reimplementing source code over reusing it.

RQ4: Is code license a barrier for code reuse for developer?
A 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 for this low percentage is that developers might not attribute the Q&A platforms from which a source code snippet comes (even though required by its license). Another possible reason is that some Q&A platforms (e.g., Stack Overflow) have a relatively restrictive code license, which might hinder developers from reusing source code in their own projects. Hence, in this RQ, we survey developers about their knowledge and understanding

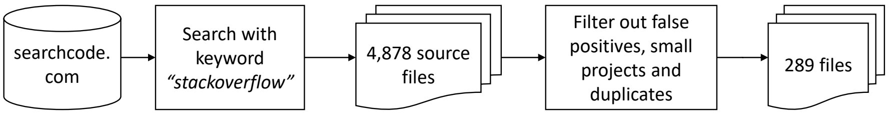


Fig. 2: An overview of our data collection of the exploratory study.

of the code licenses of Q&A platforms, to find out whether code license forms a barrier for code reuse from Q&A platforms for developers.

RQ5: How can code reuse be improved in next-generation Q&A platforms?
In this RQ, we elicit suggestions from the surveyed developers for improving next-generation Q&A platforms. We analyze and synthesize their suggestions to define a roadmap for researchers and developers of next-generation Q&A platforms.

3.2 Data Collection

In the remainder of this section, we describe our data collection process for the exploratory study (to answer RQ1 and RQ2) and our survey (RQ3–RQ5). All our studied data and the corresponding analysis are available from our online appendix (Wu *et al.*, 2017).

3.2.1 Collecting Data for the Exploratory Study

The steps of our data collection process for our exploratory study on code reuse from Stack Overflow in open-source projects (Section 4) are shown in Figure 2. First, we collected source files that contain an explicit reference to a Stack Overflow post from `searchcode.com` (Searchcode, 2016b), a source code search website. Then, we removed irrelevant files such as false positives and duplicate files.

Collecting Source Code Files from Open-Source Project Repositories: As explained in Section 2.4, developers must cite a Stack Overflow post when they reuse code or ideas from that post. Hence, to obtain source files that contain a source code snippet that is reused from a Stack Overflow post (either a question or an answer), 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.com` has indexed over 20 billion lines of source code from 7 million open-source projects. With its API (Searchcode, 2016a), we were able to collect 4,878 files in total using *"stackoverflow"* as the search keyword. We focused on files that are written in the five most popular

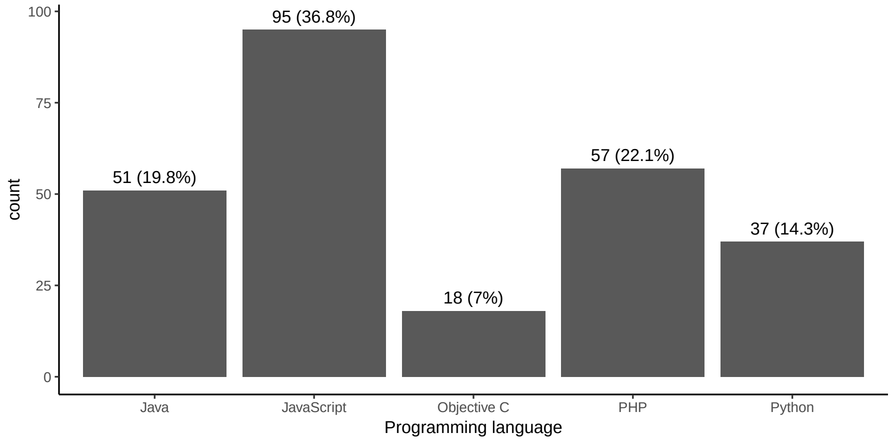


Fig. 3: The distribution of the studied Stack Overflow links over the five programming languages.

programming languages on Stack Overflow (Stack Overflow, 2016) (JavaScript, Python, Java, PHP, and Objective-C).

Removing Irrelevant Files: The *"stackoverflow"* keyword can match source files that contain the *"stackoverflow"* keyword outside of a link to a post, e.g., in an API name. We manually removed such false positives in this step. In addition, not all projects in open-source repositories are interesting from a software engineering point of view. For example, GitHub contains many toy projects, from which we cannot extract knowledge that is representative of other projects (Kalliamvakou *et al.*, 2014a). Therefore, to mitigate the effects from small projects, we removed files that belong to projects with less than 1,000 commits and 10 contributors. We then removed the files that are duplicates of each other (e.g., because they come from forked projects). Finally, we ended up with 289 unique files, which belong to 182 open-source projects. Within these files, 321 Stack Overflow hyperlinks were found. Figure 3 shows the distribution of the studied Stack Overflow links over the five studied programming languages.

3.2.2 Collecting Participants for our Survey

We used the dataset provided by Vasilescu *et al.* (2013) to get candidate participants. This dataset includes 93,771 email addresses from 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 did not exist any more. In the end, we received 453 responses which equals a response rate of 11.1%.

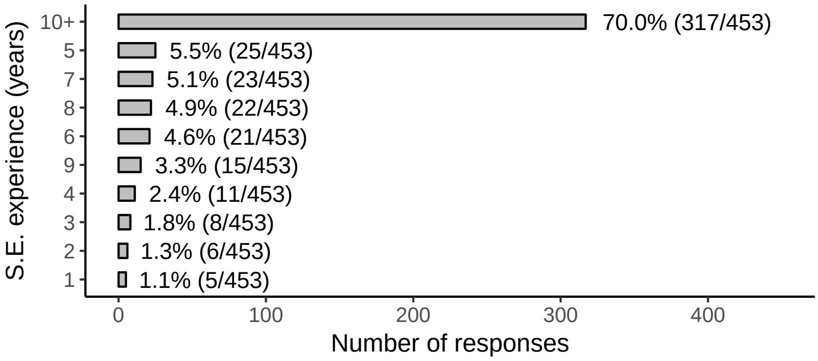


Fig. 4: Distribution of the software engineering experience of the participants in years.

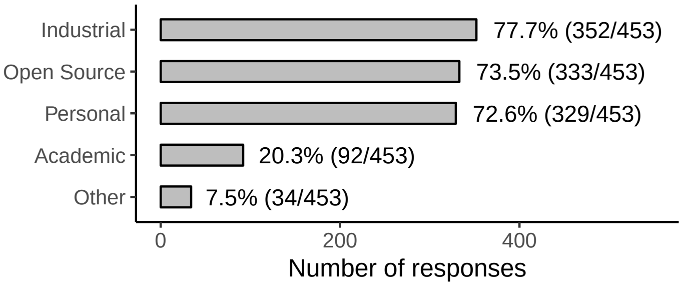


Fig. 5: Distribution of the types of projects that the participants are working on.

Figure 4 shows that 87.9% of the participants are experienced software engineers with more than 5 years experience. Industrial, open-source, and personal projects are the dominant project types that the participants are involved in, followed by academic projects (see Figure 5). Note that a participant can work on more than one type of project.

4 An Exploratory Study of Source Code Reuse from Stack Overflow in Open-Source Projects

In this section, we present and discuss the results of our exploratory study of 321 Stack Overflow links in 289 source code files of 182 open-source projects. For each research question in our exploratory study, we discuss the used approach and results.

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

Approach: To understand how developers utilize source code from Stack Overflow, we manually analyzed the collected source code and the referenced Stack Overflow posts. We manually extracted and categorized the type of code utilization from Stack Overflow posts for each collected source file. We performed a lightweight open coding-like process (Seaman, 1999; Seaman *et al.*, 2008) for identifying the type of the code utilization. This process involved 3 phases and was performed by the first three authors (i.e., P1–P3) of this paper:

- Phase I: P1 extracted a draft list of types of source code utilization from Stack Overflow based on 50 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 were revised and refined. At the end of this phase, we obtained five types of source code utilization.
- Phase II: P1 and P2 applied the resulting types of Phase I to independently categorize all 289 collected source files. They took notes regarding the deficiency or ambiguity of the types for categorizing certain source files.
- Phase III: P1, P2, and P3 discussed the coding results obtained in Phase II to revolve the disagreements until a consensus was reached. No new types were added during this discussion. The inter-rater agreement of this coding process had a Cohen's kappa of 0.91.

Table 1 shows the final categorization of the types of source code utilization from Stack Overflow. In our study, one source code file-Stack Overflow post pair could only be categorized as one type. We did not run into conflicts because of this limitation.

Results: **31.5% of the reused source code was modified in one way or another.** Table 1 shows that 20.5% of the studied files reused source code without modification (C1). In 31.5% (C2 and C3) of the files, the source code required modification before it could be used. Type C4 (12.5%) indicates that it is not exceptional that developers converted the ideas written in natural language to source code from scratch. Type C5 (35.5%) indicates that there exist developers who use Stack Overflow as a "programming manual". The finding that 31.5% of the source code reuse required additional modification 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 them, such as making the source code work in the required context. Prior studies have addressed the problem of integrating source code in a target context automatically (Feldthaus and Møller, 2013; Alnusair *et al.*, 2016; Wang *et al.*, 2016b). Our findings provide empirical support for the importance of such studies, and suggest that it may be promising to integrate the proposed code integration techniques into Q&A platforms.

Table 1: The identified types of source code utilization from Stack Overflow.

ID	Name	Definition	Count	Perc.
C1	Exact Copy	Developers copy-and-pasted source code from Stack Overflow **without any modification**.	66	20.5%
C2	Cosmetic Modification	Developers copy-and-pasted 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).	32	10.0%
C3	Non-cosmetic Modification	Developers copy-and-pasted source code from Stack Overflow with modifications which **alter the functionality** of that source code (e.g., adding arguments to a function prototype).	69	21.5%
C4	Converting Ideas	Developers did not copy-and-paste any source code from Stack Overflow. Instead, they wrote the source code from scratch by **applying the ideas** in the answers.	40	12.5%
C5	Providing Information	Developers did not reuse any source code from Stack Overflow. Instead, they treated the Stack Overflow post as an **information source** related to the issue they are addressing.	114	35.5%

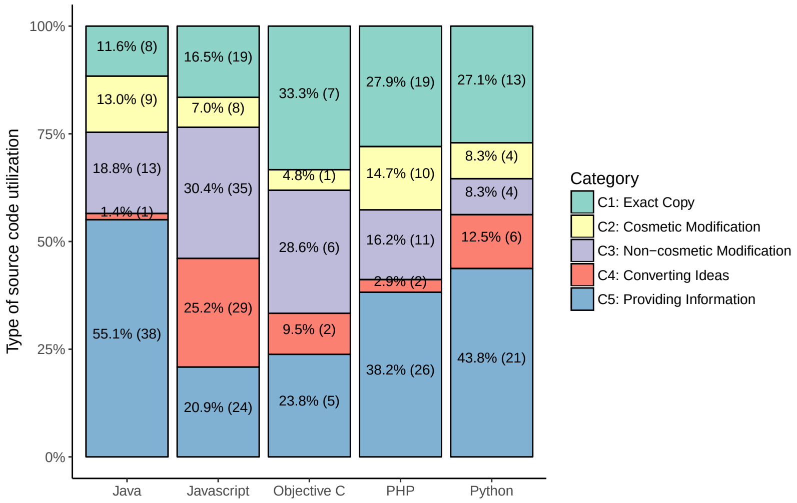


Fig. 6: The distribution of each type of source code utilization for each of the studied programming languages.

Listing 2: Code snippet from the project.[3]

```
hours = TimeUnit.MILLISECONDS.toHours(elapsedTimeMilliseconds)
    ↪ ;
minutes = TimeUnit.MILLISECONDS
  .toMinutes(elapsedTimeMilliseconds - TimeUnit.HOURS.toMillis
      ↪ (hours));
seconds = TimeUnit.MILLISECONDS
  .toSeconds(elapsedTimeMilliseconds - TimeUnit.HOURS.toMillis
      ↪ (hours)
  - TimeUnit.MINUTES.toMillis(minutes));
```

Listing 3: Code snippet from the Stack Overflow answer.[4]

```

final long hr = TimeUnit.MILLISECONDS.toHours(l);
final long min = TimeUnit.MILLISECONDS
  .toMinutes(l - TimeUnit.HOURS.toMillis(hr));
final long sec = TimeUnit.MILLISECONDS
  .toSeconds(l - TimeUnit.HOURS.toMillis(hr)
  - TimeUnit.MINUTES.toMillis(min));
```

In 10.0% 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 Listing 2 and 3, the developer copied three lines of source code in the accepted answer from the Stack Overflow post and renamed the variable name from `hr`, `min`, and `sec` to `hours`, `minutes`, and `seconds`, respectively.

In 12.5% of the files, developers wrote the source code from scratch based on the descriptions of the algorithm. In the example shown in Listing 4 and Listing 5, developers implemented a function to detect whether a line intersects with a rectangle (Listing 4), based on an answer from the Stack Overflow post shown in Listing 5.

Another example is shown in Listing 6 and 7, where the developers wrote a regular expression that extracts all Youtube video ids in a string (see Listing 6). This source code snippet was modified based on the source code from the Stack Overflow post shown in Listing 7, which was written in PHP. In this example, developers actually rewrote the regular expression in JavaScript

[3] https://goo.gl/9ouSz1

[4] https://goo.gl/74oVBu

[5] https://goo.gl/4ezMUr

[6] https://goo.gl/1Wn9vF

[7] https://goo.gl/HK5kyV

[8] https://goo.gl/eq1Dnk

Listing 4: Type C5: An example of converting descriptions into source code: code snippet from the project[5] is implemented based on the description of the algorithm from Stack Overflow (see Listing 5).

```
Rect.prototype.collideLine = function(p1, p2) {
  var x1 = p1[0];
  var y1 = p1[1];
  var x2 = p2[0];
  var y2 = p2[1];

  function linePosition(point) {
    var x = point[0];
    var y = point[1];
    return (y2-y1)*x + (x1-x2)*y + (x2*y1-x1*y2);
  }

  var relPoses = [[this.left, this.top],
             [this.left, this.bottom],
             [this.right, this.top],
             [this.right, this.bottom]
            ].map(linePosition);

  var noNegative = true;
  var noPositive = true;
  var noZero = true;
  relPoses.forEach(function(relPos) {
    if (relPos > 0) {
      noPositive = false;
    } else if (relPos < 0) {
      noNegative = false;
    } else if (relPos === 0) {
      noZero = false;
    }
  }, this);

  if ( (noNegative || noPositive) && noZero) {
    return false;
  }
  return !((x1 > this.right && x2 > this.right) ||
        (x1 < this.left && x2 < this.left) ||
        (y1 < this.top && y2 < this.top) ||
        (y1 > this.bottom && y2 > this.bottom)
        );
};
```

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 used Stack Overflow posts in 35.5% of the files as an information source for later reference. In 35.5% of the files, developers did not reuse any source code from Stack Overflow. Instead, they put a Stack

Listing 5: Description of the algorithm in the Stack Overflow answer.[6]

```
Let the segment endpoints be p1=(x1 y1) and p2=(x2 y2).
Let the rectangle's corners be (xBL yBL) and (xTR yTR).

Then all you have to do is

A. Check if all four corners of the rectangle are on the
same side of the line. The implicit equation for a line
through p1 and p2 is:

F(x y) = (y2-y1)x + (x1-x2)y + (x2*y1-x1*y2)

If F(x y) = 0, (x y) is ON the line.
If F(x y) > 0, (x y) is "above" the line.
If F(x y) < 0, (x y) is "below" the line.

Substitute all four corners into F(x y). If they're all
negative or all positive, there is no intersection. If
some are positive and some negative, go to step B.

B. Project the endpoint onto the x axis, and check if the
segment's shadow intersects the polygon's shadow. Repeat
on the y axis:

If (x1 > xTR and x2 > xTR), no intersection (line is to
right of rectangle).
If (x1 < xBL and x2 < xBL), no intersection (line is to
left of rectangle).
If (y1 > yTR and y2 > yTR), no intersection (line is
above rectangle).
If (y1 < yBL and y2 < yBL), no intersection (line is
below rectangle).
else, there is an intersection. Do Cohen-Sutherland or
whatever code was mentioned in the other answers to
your question.

You can, of course, do B first, then A.
```

Listing 6: Code snippet from the project in JavaScript.[7]

```
YOUTUBE_REGEXP: new RegExp(
  '(?:https?://)?' + // Optional scheme. Either...
  '(?:www\\.)?'    + // Optional www subdomain
  '(?:'            + // Group host alternatives
  'youtu\\.be/'    + // Either youtu.be,
  [...]
  ')'        // End negative lookahead assertion.
),
```

Listing 7: Code snippet from the Stack Overflow answer in PHP.[8]

```
//Linkify youtube URLs which are not already links
function linkifyYouTubeURLs($text) {
  $text = preg_replace('~(?#!js YouTubeId Rev:...
    # Match non-linked youtube URL in the wild...
    https?://            # Required scheme...
    (?:[0-9A-Z-]+\.)?    # Optional subdomain.
    (?:                  # Group host alternatives.
    youtu\.be/           # Either youtu.be,
    [...]
  $text);
  return $text;
}
```

Overflow hyperlink in their source code to provide background information about the issue or solution. For example, there is a file[9] in which the developer gave a warning that the usage of dict can be dangerous if multiple headers are set in the Set-Cookie header and the developer also provided the link to the Stack Overflow post which discussed this issue in the source code.

Developers are the most likely to reuse code or ideas in JavaScript. Figure 6 shows the distribution of each type of source code utilization for each studied programming language. We observe that code and idea reuse was the highest in JavaScript (79.1% of the studied JavaScript files). One possible explanation is that Stack Overflow provides an online running environment for JavaScript, which may make developers more confident about reusing code or ideas in JavaScript from Stack Overflow than in other languages.

31.5% of the reused source code required additional modification, which shows the importance of studies on automatic code integration. In 12.5% of the studied files, developers reimplemented code based on an idea, which suggests that Q&A platforms should consider to summarize the key points that are discussed in a post to give developers a quick view of the question and its answers.

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

Approach: We manually inspected 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-scored 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. The discrepancies were due to the difficulty of identifying the

[9] https://goo.gl/KKbPWk

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

Source	Highest-voted	Non-highest-voted	Total	Perc.
Accepted Answer	144	11	155	48%
Non-Accepted Answer	35	48	83	26%
Question	-	-	5	2%
NOT REUSE	-	-	78	24%
Total	-	-	321	100%

Listing 8: Code snippet in the project that implements a method to generate GUIDs.[10]

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

exact answer that was reused (in particular, when only the idea of a code snippet was reused). After identifying the reused answer, the categorization was straightforward. The inter-rater agreement of this categorization had a Cohen's kappa of 0.85.

Results: **In 26% of the studied files developers chose a non-accepted answer and in 58%, those non-accepted answers were not the highest-scored ones.** The results of the categorization are shown in Table 2. As we can see from the results, not all reused source code came from an accepted answer. In 48% of the studied files, developers chose source code from an accepted answer. However, there are still a considerable number (26%) of files where developers choose the source code from non-accepted answers. Moreover, among those non-accepted answers, 58% were not the highest-scored ones, which indicates that developers certainly did not always choose source code from the accepted or highest-scored answer. In the remainder of this section, we discuss the situations in which developers reused source code from a non-accepted answer in more detail.

4.2.1 Different Requirements than the Question Asker

Description: Developers chose source code from a non-accepted answer because they had different requirements than the original question asker.

Example: A developer wanted to implement a method to generate GUIDs. The source code in this example is shown in Listing 8. This source code snippet is

[10] `https://goo.gl/Z1pRMS`

Listing 9: Code snippet in the accepted answer on Stack Overflow.[12]

```
function guid() {
  function s4() {
    return Math.floor((1 + Math.random())
      * 0x10000).toString(16)
      .substring(1);
  }
  return s4() + s4() + ’-’ + s4() + ’-’ + s4()
    + ’-’ +s4() + ’-’ + s4() + s4() + s4();
}
```

actually from a non-accepted answer[11] on Stack Overflow which has 37 votes, while the accepted answer has 1290 votes. The source code provided by the accepted answer is shown in Listing 9.

According to the description in the answer that contains the source code, the algorithm in Listing 8 is simpler and has very good performance, but not compliant with the RFC 4122 standard. The author of this answer also attached a performance test result in which several algorithms that are mentioned in other answers of the Stack Overflow post are compared, which shows that the algorithm in Listing 8 outperforms the others. Hence, one possible explanation is that the developer who adopted this low-scored answer prioritizes performance and simplicity over other factors, such as whether the generated result is compliant with a standard.

4.2.2 Fixing Bugs

Description: Developers adopted source code that improves on the accepted answer (e.g., by fixing a bug or handling additional cases).

Example: A developer was looking for a method to draw a dashed line around a selection area in JavaScript[13]. The non-accepted answer[14] improves the accepted answer by utilizing the 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: Developers adopted source code with a better performance.

Example: A developer was looking for an algorithm that sorts an array by the Levenshtein Distance in JavaScript. According to the comments below the

[11] https://goo.gl/aC4auZ

[12] https://goo.gl/xpAcga

[13] https://goo.gl/gzMCgy

[14] https://goo.gl/8foVXq

accepted answer, the implementation in the accepted answer performed better than the one provided by the original asker. However, a non-accepted answer provided 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 this developers gave performance a higher priority.

Unsurprisingly, we found that developers have different requirements for their solutions. Even if answers that are provided in the post do not meet the requirements of the asker, other developers may find them useful (e.g., a 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 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 confirm the need for this improvement (see Section 6.2).

Developers reused code from a non-accepted or low-scored answer for various reasons, such as the simplicity and performance of the source code. Some even reused code from an answer that delivered a total opposite from what the asker wanted. Hence, Q&A platforms should improve the way in which answers are organized, so that developers can find the most suitable answers based on their requirements easily.

5 A Survey on Code Reuse from Stack Overflow

Survey Design: Two of the authors posited the survey questions that cover the three research questions (see Section 3.1). The third author checked the questions to eliminate any ambiguity from the wording of the survey. Before sending the survey to the 6000 participants that we collected in Section 3.2.2, we sent a draft version of the survey to 20 participants (excluded from the 6000 participants). We received feedback from seven of them, and refined the survey based on this feedback. The questions in the survey are available in the Appendix. The survey is divided into three parts:

1. **Demography (Q1 - Q7)**: these questions collect information about the software engineering background of the participants.
2. **Barriers (Q8 - Q17)**: these questions collect information about the barriers that the participants face when reusing source code from Q&A platforms. We included only the responses from participants who have ever reused source code from Q&A platforms (i.e., those who answered *yes* to Q7, which were 380 (83.9%) participants).
3. **Suggestions (Q18 - Q19)**: these questions collect suggestions for next generation Q&A platforms. Every participant could answer these two ques-

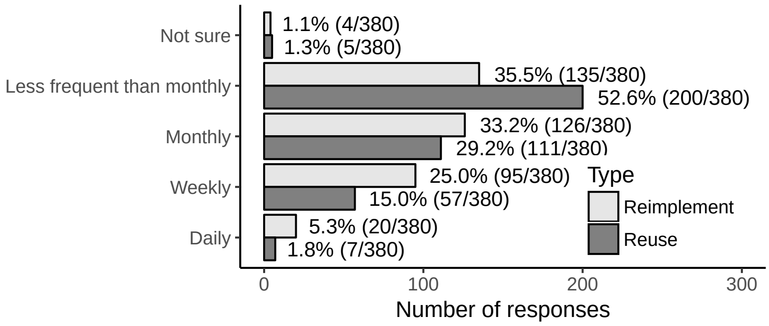


Fig. 7: Comparison of frequency of reusing and reimplementing source code.

tions, regardless of whether they ever reused source code from Q&A platforms.

Data Analysis: The responses of the survey are available in our online appendix (Wu *et al.*, 2017). The survey contained 12 open-ended questions in which participants could choose to input their own responses in free-form text. For each of these questions, we used an open coding-like approach to let the coding schema emerge during the analysis (Glaser, 2017). We adopted a three phase coding process:

- Phase I: two of the authors coded the responses of each open-ended question individually. As a result, both authors had their 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.
- Phase II: the two authors used the revised schema to code the answers. Then, they discussed and resolved conflicts. The Cohen's Kappa value for this coding result was 0.92. As a result, a unified coding schema was developed and applied to all the answers.
- Phase III: three of the authors discussed the coding results obtained in Phase II to revolve disagreements until a consensus was reached. The interrater agreement of this coding process had a Cohen's kappa of 0.79.

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

5.1 **RQ3: What are the preferences of developers when it comes to reusing code?**

Developers reimplement source code slightly more frequently (i.e., for daily, weekly, and monthly cases) than that they reuse source code. Figure 7 shows the comparison of frequency of reimplementing source code and reusing source code from Q&A platforms. The number of participants

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%

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.

A majority of developers (65%) prefer reimplementing source code, due to the code modification that is required to make the code from the post work in their own project. Table 3 shows the reasons for choosing reimplementation over the reuse of source code. The top reason that makes developers prefer reimplementing 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 RQ1 (i.e., most code needs modification before reusing) and also provides empirical evidence for the importance of research on code integration. Several studies have been done on automatically retrieving and integrating code in a user's project context (Alnusair *et al.*, 2016; Galenson *et al.*, 2014; Wang *et al.*, 2016b). However, these approaches are not widely adopted by developers. Future studies should investigate which factors prevent such tools from being applied in practice.

Code comprehension ranks as the second most important reason that preferring reimplementation over code reuse. This finding is in line with the work by Xin *et al.* (2017), which showed that developers spend 58% of their time on program comprehension activities. Hence, next-generation Q&A platforms should investigate how to improve comprehension of the source code in a post to facilitate its reuse. Approximately one-third (32%) of the participants complained about the low code quality on Stack Overflow, which highlights the need for next-generation Q&A platforms to improve or verify the code quality of source code snippets.

An interesting observation was that 17% of the participants stated 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 snippet is large or complex, it could take more time to comprehend it than to make it work in another context.

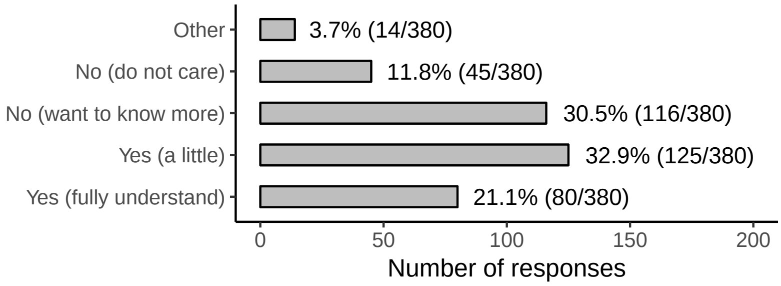


Fig. 8: Participants' awareness of the licenses of Q&A platforms.

Developers reimplement source code slightly more frequently than that they reuse source code. The primary reason is that it would take longer to adapt the source code to work in their own projects, than to simply reimplement it. Our observations provide empirical evidence for the importance of research on automated code integration and code comprehension, and highlight the need of improving the quality of code snippets on next-generation Q&A platforms.

5.2 **RQ4: Is code license a barrier for code reuse for developers?**

In 75.2% of the cases, participants do not have a good understanding of the license terms of Q&A platforms, which indicates that there may be license violation issues when developers reuse source code from Q&A platforms. Figure 8 shows the results of participants' awareness of the licenses of Q&A platforms. An *et al.* (2017) studied code reuse on Android apps and observed 1,279 potential license violation cases where developers reused source code from Q&A platforms in Android apps, or vice versa. Our survey results give a possible explanation for such violations. The *"Other"* category in Figure 8 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."*

In 39.2% of the cases, participants are not sure whether the license of a Q&A platform is compatible with that of their own project. Figure 9 shows that an additional 12.1% of the participants (strongly) disagrees that the license of a Q&A platform is compatible with that of their own project. Hence, 51.3% of the participants may experience difficulties (or cause a violation) when reusing code from Q&A platforms due to their license. These difficulties were noticeable from the survey responses when the participants were asked why they preferred reimplementing over reusing source code. For example, one participant mentioned that *"licensing is sometimes an issue."*.

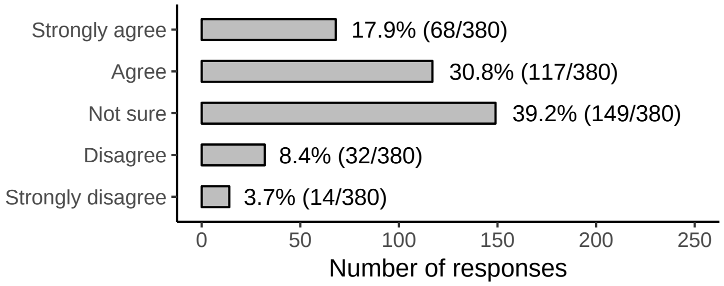


Fig. 9: Participants' opinion about license compatibility between Q&A platforms and their projects.

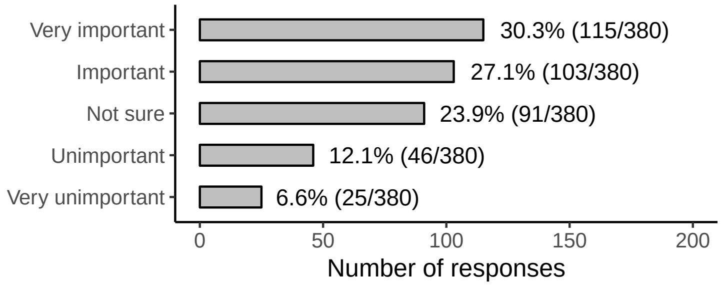


Fig. 10: Importance of having more information on license.

More than half of the participants (57.4%) think that having more information about the code license is (very) important (see Figure 10). Together with the finding that most participants do not have a good understanding of the code license of Q&A platforms, these findings reveal a need for clearer information about the code license of Q&A platforms.

Generally speaking, participants did not have a good understanding of the code license on Q&A platforms. More than half of the participants believed that, or were unsure whether, there exist incompatibilities between the code license of their own project and Q&A platforms. Almost 60% of the participants thought that Q&A platforms should give more information about their code license. Based on these observations, next-generation Q&A platforms should have clearer license information and make that information more visible to developers.

6 A Roadmap for Next-Generation Q&A platforms

In this section, we summarize and analyze the results for RQ5 (How can code reuse be improved in next-generation Q&A platforms?). In total, we collected

Table 4: The categorization of code quality suggestions — 64 out of 183 (35.0%).

Category	Description (D) – Example (E)	Perc.	Count
Integrated validator	**D:** Integrated validator that can test the code snippets on Q&A platforms. **E:** *"An inbuilt REPL environment for as many languages/environments as possible."*	42.2%	27
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:** *"make date important in marking outdated code, and deprecate those snippets via the community"*	29.7%	19
Answer quality	**D:** Classifier that helps distinguish high and low quality answers. **E:** *"Better support for answers that are good, but out of date."*	17.2%	11
Code review	**D:** Integrated code review tool that helps improve the code quality. **E:** *"In-browser code review and commenting similar to that provided by commercial code review tools."*	10.9%	7

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. In total, we extracted 183 suggestions from the responses. Using our open coding-like approach (see Section 5), each suggestion was categorized into one of these five categories: code quality, information enhancement & management, data organization, license, and human factors. We categorized suggestions that did not fall into one of these categories into a sixth "other" category. We highlight the findings and discuss implications on future research of next-generation Q&A platforms of each category in the remainder of this section.

6.1 Suggestions on Code Quality

Next-generation Q&A platforms should integrate mechanisms for online code validation and detecting outdated code. Code quality is the most popular type of suggestion (35.0%) from the participants. Table 4 shows the categorization of the suggestions that participants made on improving the code quality on Q&A platforms. The two most important suggestions from developers on improving code quality were adding (1) an integrated validator (27 participants) that can test source code online and (2) an outdated code detection mechanism (19 participants) that can identify code for old software versions.

An integrated validator is a convenient way of testing source code snippets online to ensure the quality of the source code. Participants described such a tool for example as follows: *"The ability to interact with and run the code examples written in answers and questions".* Several Q&A platforms have started to integrate online validation into their websites. For example, Stack Overflow can validate three web-languages: HTML, CSS, and JavaScript (Stack Overflow, 2014). However, Stack Overflow does not support online validation of other languages, such as Java and C++, which are also very popular on Stack Overflow. There are several challenges when it comes to online validation of all languages.

One of the biggest challenges is to make an incomplete code snippet run correctly, since code snippets on Q&A platforms are usually not minimal working examples (MWEs). Often the answerer only needs to implement the core part of a solution and may leave out necessary context information (e.g., the required software version). To address this problem, 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, it is difficult to automatically infer the exact library based on the source code snippet only. 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 automatic code extension.

Providing an outdated code detection mechanism is the second most popular suggestion in the code quality category. Many of the participants mentioned that source code on Q&A platforms is often outdated and not suitable for current technologies or situations. For example, one participant suggested: *"Have explicit mechanisms for dealing with content that goes out of date due to platform or language changes."* Some participants suggested 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 suggested deprecating 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.

However, as far as we know, no existing study has investigated outdated source code or solutions on Q&A platforms so far. Hence, there is a need for future research on developing mechanisms to deal with outdated source code or solutions. There are two primary directions to deal with outdated code or solutions. First, future studies should propose approaches to automatically identify *outdated source code* or solutions in Q&A platforms using machine

Table 5: The categorization of Information enhancement & management suggestions — 43 out of 183 (23.5%).

Category	**Description (D) – Example (E)**	**Perc.**	**Count**
Answer tagging	**D:** Better tagging-like information system for answers. **E:** *"Provide/require tagging of the version number(s) of the language [...]"*	37.2%	16
Code evolution	**D:** Better management of the evolution/revisions of code snippets. **E:** *"where does the code come from and copied to, and also the revisions inside the platform."*	14.0%	6
Resources linking	**D:** Q&A platforms should suggest for other resources (e.g., books, API documents, libraries etc.) **E:** *"Books suggestions based on questions."*	11.6%	5
Answer writing support	**D:** Support for writing better questions/answers. **E:** *"[...] it would be nice if it would be easier to ask a good question [...]"*	9.3%	4
Other	**D:** Other aspects of information enhancement & management. **E:** *"Built in support within an IDE to make it faster to get the answer you are interested in."*	27.9%	12

learning techniques. Second, future research should investigate how *incentive systems* can motivate communities to identify and update outdated source code or solutions.

6.2 Suggestions on Information Enhancement & Management

Next-generation Q&A platforms should allow tagging for answers. Information enhancement & management (23.5%) is the second most popular suggestion for developers on Q&A platforms. Table 5 presents the results of suggestions related to information enhancement and management. Based on the observations, future studies should focus on recommending the tagging-like information for answers. These recommendations could be 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 Suggestions on Data Organization

Next-generation Q&A platforms should better organize their data, for example by providing a better searching and indexing mechanism

Table 6: The categorization of data organization suggestions — 21 out of 183 (11.5%).

Category	Description (D) – Example (E)	Perc.	Count
Code searching/indexing	**D:** Support for easier code search. **E:** *"Source Code indexing for easier retrieval. It could also give the possibility to find example of usage functions."*	47.6%	10
Duplicate posts	**D:** An automatic way of clustering duplicate questions/answers. **E:** *"Auto-suggest similar questions, particularly for questions that don't have answers."*	38.1%	8
Comments	**D:** Support on utilizing the comments of posts. **E:** *"Code in *comments* must be expressed better, than on Stack Overflow."*	14.3%	3

and better duplicate detection. As shown in Table 6, ten participants suggested that Q&A platforms should have a better way to index and search code. For example, one of the participants mentioned: *"Ability to search questions based on the version of the framework or language I'm working with"*. Eight participants suggested that Q&A platforms should have an automatic way to detect duplicate or similar posts and be organized in a better way. Three participants suggested to improve the utilization of the comments on posts.

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

Researchers proposed various approaches to help Q&A platforms detect duplicate questions automatically (Zhang *et al.*, 2015; Ahasanuzzaman *et al.*, 2016; Wang *et al.*, 2017d; Zhang *et al.*, 2017). The common way to identify duplicate questions is to measure such questions' similarity in terms of semantic meaning. Recently, deep learning has proven its power of capturing semantic meaning from natural language in several studies (Ganguly *et al.*, 2015; Lai *et al.*, 2015; Bian *et al.*, 2014; Chen *et al.*, 2016). Hence, future research could consider to employ deep learning to detect duplicate or find similar questions.

6.4 Suggestions on Code License

Next-generation Q&A platforms should make their code licensing information clearer and more visible. In 12.6% of the cases, participants suggested to improve license-related issues, in particular to make the license more clear (16 participants). This percentage is in line with our earlier finding that 75.2% of the participants did not have a good understanding of the license

Table 7: The categorization of code license suggestions — 23 out of 183 (12.6%).

Category	Description (D) – Example (E)	Perc.	Count
Clearer license	**D:** Q&A platforms should make their license terms clearer. **E:** *"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."*	69.6%	16
Permissive license	**D:** Q&A platforms should use a more permissive license. **E:** *"Let the user choose a more re-user-friendly license (e.g. copy without reference)."*	30.4%	7

terms of Q&A platforms (see Section 5.2). Table 7 shows the suggestions about the code license of Q&A platforms.

Participants requested that Q&A platforms provide a clearer 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 would neglect the license of Q&A platforms and reuse source code from these platforms, they are under the risk of license violation which may cause legal problems later. The scale of license violation has been studied by An *et al.* (2017). An et al. investigated code reuse in Android apps and observed a significant number of potential license violation cases when developers reused source code from Q&A platforms in Android apps, and vice versa.

Seven participants suggested that Q&A platforms should use a more permissive license, which has fewer 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 to 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 the CC BY-SA 3.0 license 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 a post on Stack Exchange (Stack Exchange, 2015), the change is not reflected on their homepage (Stack Overflow, 2017), which still says *"user contributions licensed under cc by-sa 3.0 with attribution required"*. As such mismatches will further deepen developers' misunderstanding of license terms. Therefore, we suggest that next-generation Q&A platforms explicitly describe their license terms for source code reuse in a consistent manner.

Table 8: The categorization of human factor suggestions — 19 out of 183 (10.4%).

Category	Description (D) – Example (E)	Perc.	Count
Better curator	**D:** Better curators are needed to help improve the quality of the posts. **E:** *"Definitely curators for specific languages to rate answers in specific areas."*	63.2%	12
Gamification-related	**D:** Suggestions on improving the gamification system of Q&A platforms. **E:** *"Base reputation on number of answers up-voted by others, not on personal activity."*	36.8%	7

6.5 Suggestions on the Human Factor

Next-generation Q&A platforms should assign human experts to curate knowledge on the platform. In 10.4% of the cases, participants suggested to improve Q&A platforms in terms of the human factor. Table 8 shows that twelve participants suggested 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 emphasized the importance of collaboration within 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."* Zagalsky *et al.* (2017) revealed a shift within the R community from knowledge creation to knowledge curation. Hence, we suggest that next-generation Q&A platforms study other communities to improve the knowledge curation and collaboration processes.

Seven participants suggested 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 several 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.)"*

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

6.6 Other Suggestions

Table 9 shows the 13 suggestions that did not fall into the other categories. There were four suggestions that suggested that Q&A platforms are open-sourced. In addition, there were three suggestions that were related to AI techniques. For example, one of them suggested the use of an AI technique that can automatically produce source code while developers only need to

Table 9: The categorization of other suggestions — 13 out of 183 (7.0%).

Category	Description (D) – Example (E)	Perc.	Count
Open source	**D:** Open sourced/community-hosted Q&A platforms. **E:** *"Open source to be competitive, self-hosted and easy to deploy (even without requiring docker or similar, to be usable in low end containers or even in hosting platforms)."*	30.8%	4
AI	**D:** Include AI-related techniques into the Q&A platforms. **E:** *"Let AI write code, we do code review."*	23.0%	3
Other	**D:** Cross-language support. Q&A platforms. **E:** *"When the clone is written in a different language, a language translation is automatically performed if not with some manual assistance"*	46.2%	6

review the source code. These suggestions support the current research on code generation, in which tools are developed to automatically generate code based on a natural language input (Yin and Neubig, 2017; Gu *et al.*, 2016). The rest of the suggestions in the "other" category talk about different topics. For example, one of them suggests the integration of a tool that can automatically translate a source code snippet across programming languages.

In summary, based on the findings from our survey, we observe that the difficulty in fitting code in their own projects, a low comprehension and low quality of the code are the top barriers that prevent developers from reusing code. Lacking a good understanding of the code license is also a significant barrier for code reuse. Thus, future studies are encouraged to address these barriers to better facilitate code reuse for developers.

7 Threats to Validity

External validity. Threats to external validity relate to the generalizability of our findings. In this study, we used `searchcode.com` as our search engine and found 4,878 source files in total that contained links to Stack Overflow posts. After removing the small projects and duplicate files, there were 321 Stack Overfow links and 289 files left. The number of files may not be large enough to represent all the cases in the real world. The reason for restricting our study to these 321 Stack Overflow links is that we want to ensure that we study code reuse in serious software projects only. Kalliamvakou *et al.* (2014b) showed that many open source projects are toy projects which do not adequately reflect software engineering practices. Hence, when studying open source projects, it is important to ensure those toy projects are removed from the data set. We ensure that such projects are removed by imposing strict selection criteria on our data (i.e., we only study files from projects that have at least 1000 commits and 10 contributors). We are confident that the studied number is large enough for our exploratory study to identify the core issues that are involved in the process of code reuse from Q&A platforms.

For the survey, we only invited developers who are in the intersection of users of GitHub and Stack Overflow. Hence, our results may not generalize to software developers who are not in this intersection, such as developers of closed-source software. Future studies should extend our exploratory study and survey to developers from other domains.

We focused the first part of our study on the five most popular programming languages, and as a result, our findings may not generalize to other languages. Future work is necessary to investigate whether our findings hold for other languages.

Most of our findings are general in nature and seem to apply to code reuse in general. However, as we have no solid evidence, we cannot make definite claims about the generalizability of our findings outside Q&A platforms. Future studies should investigate whether our findings are valid outside Q&A platforms.

Internal validity. Threats to internal validity relate to the experimenter bias and errors. In this paper, we heavily rely on manual analysis. For example, we manually inspected the source code in the projects and in Stack Overflow posts, we manually categorized each case of how developers are reusing the source code, and we manually categorized the survey responses. Unfortunately, these tasks are extremely difficult to automate. For example, it is very hard to automatically identify the Non-cosmetic Modification, Converting Ideas, and Providing Information types. Latent Dirichlet allocation (LDA) will not work because LDA heavily relies on the similarity of the used terminology in two files. In the aforementioned types, the used terminology across two files tends to be different. Other techniques, such as clone detection techniques cannot capture the Non-cosmetic Modification, Converting Ideas, and Providing Information types either. Hence, we had no option other than to perform our analysis in a manual fashion. To mitigate the threat of bias during the manual analysis, two of the authors conducted the manual analysis and discussed any conflicts with a third author until a consensus was reached. We used Cohen's kappa (Gwet *et al.*, 2002) to measure the inter-rater agreement. The kappa values ranged from 0.79 to 0.92, which implies a high level of agreement. In addition, to improve the replicability of our study, we made our studied data and results (including the coding results of our exploratory study and survey) available in our online appendix Wu *et al.* (2017).

Another threat is that our findings in RQ2 may not exactly reflect the intent of a user who reused code from Stack Overflow, since the code provided in the post to which the link points may not exactly be what the developer needs. To mitigate this threat, we tried contacting the developers about their intent, but received no response.

Construct validity. A threat to the construct validity of this study is that we used mostly closed-ended questions in our survey, which may affect the richness of the responses collected from participants. However, open-ended questions have several disadvantages (Reja *et al.*, 2003): (1) Open-ended questions take much longer for participants to fill out, making it more likely that they do not fill out the survey at all. (2) Open-ended questions have the problem of missing data, i.e., participants skipping or doing a poor job at answering a question (i.e., by giving an incomplete or invalid answer). (3) Open-ended questions are much more difficult to code than closed-ended questions. We felt that for most of our survey, the advantages of closed-ended questions outweighed the disadvantages. Therefore, we chose to use mostly closed-ended questions, together with an "Other." field. Another threat is that Question 18 (see the appendix for details) could influence the participants' answer to Question 19. We used Question 18 as an "icebreaker", so that the participants could start their thought process from there. Future studies should consider this effect when performing user surveys.

8 Conclusion

Prior studies on code reuse from Q&A platforms have focused on locating and integrating source code in a required context automatically. However, no studies have been done on *how* developers utilize source code from Q&A platforms. In this paper, we studied how developers reuse code from Q&A platforms and we identified the barriers that they faced during the code reuse process. The most important findings of our study are:

1. Our exploratory study shows that 78.2% of the reused source code from Stack Overflow had to be modified in order to work in the required context. The required modification ranged from simple refactoring to a complete reimplementation.
2. Developers reuse source code snippets from non-accepted answers as well (26% of the studied cases).
3. Developers prefer reimplementing source code over reusing source code because of the difficulty of integrating the code into their own project, and low comprehension or low quality of the code.
4. The most suggested improvements for next-generation Q&A platforms are about improving the quality of source code snippets.
5. Many developers do not understand, or do not seem to care much about the code license of Q&A platforms.

Our study shows that Q&A platforms are evolving beyond their traditional use of asking and answering questions. From our survey, we can conclude that code reuse from Q&A platforms is a real challenge for developers: judging by the suggestions that we extracted from the survey, many developers face similar barriers during the code reuse process. Next-generation Q&A platforms should integrate existing solutions that improve the quality, comprehension and organization of source code snippets to better facilitate code reuse. Researchers can leverage the roadmap that is presented in this paper to remove many of these barriers in code reuse.

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 Hoye, 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.

Barzilay, O. (2011). Example embedding. In *Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2011, pages 137–144.

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.

Galenson, J., Reames, P., Bodik, R., Hartmann, B., and Sen, K. (2014). Codehint: Dynamic and interactive synthesis of code snippets. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 653–663.

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.

Gharehyazie, M., Ray, B., and Filkov, V. (2017). Some from here, some from there: Cross-project code reuse in github. In *Proceedings of the 14th International Conference on Mining Software Repositories*, MSR '17, pages 291–301.

Glaser, B. (2017). *Discovery of grounded theory: Strategies for qualitative research*. Routledge.

Gu, X., Zhang, H., Zhang, D., and Kim, S. (2016). Deep API learning. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*, pages 631–642. ACM.

Gwet, K. *et al.* (2002). Inter-rater reliability: dependency on trait prevalence and marginal homogeneity. *Statistical Methods for Inter-Rater Reliability Assessment Series*, **2**, 1–9.

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. (2014a). The promises and perils of mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*, pages 92–101. ACM.

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., and Damian, D. (2014b). The promises and perils of mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*, pages 92–101.

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.

Ponzanelli, L., Mocci, A., Bacchelli, A., and Lanza, M. (2014c). Understanding and classifying the quality of technical forum questions. In *Proceedings of the 14th International Conference on Quality Software (QSIC)*, pages 343–352.

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.

Reja, U., Manfreda, K. L., Hlebec, V., and Vehovar, V. (2003). Open-ended vs. close-ended questions in web questionnaires. *Developments in Applied Statistics (Metodološki zvezki)*, **19**, 159–77.

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. and Robillard, M. P. (2017). Understanding stack overflow code fragments. In *2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017, Shanghai, China, September 17-22, 2017*, pages 509–513.

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.

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, S., Chen, T.-H., and Hassan, A. E. (2017b). Understanding the factors for fast answers in technical Q&A websites. *Empirical Software Engineering*, pages 1–42.

Wang, X., Pollock, L. L., and Vijay-Shanker, K. (2014c). Automatic segmentation of method code into meaningful blocks: Design and evaluation. *Journal of Software: Evolution and Process*, **26**(1), 27–49.

Wang, X., Pollock, L. L., and Vijay-Shanker, K. (2017c). Automatically generating natural language descriptions for object-related statement sequences. In *IEEE 24th International Conference on Software Analysis, Evolution and Reengineering, SANER 2017, Klagenfurt, Austria, February 20-24, 2017*, pages 205–216.

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. (2017d). Bilateral multi-perspective matching for natural language sentences. *CoRR*, **abs/1702.03814**.

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.

Wu, Y., Wang, S., Bezemer, C.-P., and Inoue, K. (2017). Online appendix of manuscript "How Do Developers Utilize Source Code from Stack Overflow?". `https://zenodo.org/record/1116508`.

Xia, X., Bao, L., Lo, D., Kochhar, P. S., Hassan, A. E., and Xing, Z. (2017). What do developers search for on the web? *Empirical Software Engineering*.

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.

Yin, P. and Neubig, G. (2017). A syntactic neural model for general-purpose code generation. *CoRR*, **abs/1704.01696**.

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.

Zagalsky, A., German, D. M., Storey, M.-A., Teshima, C. G., and Poo-Caamaño, G. (2017). How the R community creates and curates knowledge: an extended study of Stack Overflow and mailing lists. *Empirical Software Engineering*.

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 □ Visual Basic .Net
□ JavaScript □ Assembly □ PHP □ Perl □ Ruby
□ 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?