2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM) | 978-1-6654-4897-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/SCAM52516.2021.00027

What Do Developers Discuss about Code Comments?

Pooja Rani*, Mathias Birrer*, Sebastiano Panichella†, Mohammad Ghafari‡, Oscar Nierstrasz*

*Software Composition Group, University of Bern
Bern, Switzerland
scg.unibe.ch/staff
† Zurich University of Applied Science (ZHAW), Switzerland
panc@zhaw.ch
‡ University of Auckland, New Zealand
m.ghafari@auckland.ac.nz

***Abstract*—Code comments are important for program comprehension, development, and maintenance tasks. Given the varying standards for code comments, and their unstructured or semi-structured nature, developers get easily confused (especially novice developers) about which convention(s) to follow, or what tools to use while writing code documentation. Thus, they post related questions on external online sources to seek better commenting practices. In this paper, we analyze code comment discussions on online sources such as Stack Overflow (SO) and Quora to shed some light on the questions developers ask about commenting practices. We apply Latent Dirichlet Allocation (LDA) to identify emerging topics concerning code comments. Then we manually analyze a statistically significant sample set of posts to derive a taxonomy that provides an overview of the developer questions about commenting practices.**

Our results highlight that on SO nearly 40% of the questions mention how to write or process comments in documentation tools and environments, and nearly 20% of the questions are about potential limitations and possibilities of documentation tools to add automatically and consistently more information in comments. On the other hand, on Quora, developer questions focus more on background information (35% of the questions) or asking opinions (16% of the questions) about code comments. We found that (i) not all aspects of comments are covered in coding style guidelines, *e.g.*, how to add a specific type of information, (ii) developers need support in learning the syntax and format conventions to add various types of information in comments, and (iii) developers are interested in various automated strategies for comments such as detection of bad comments, or verify comment style automatically, but lack tool support to do that.

***Index Terms*—Mining online sources, Stack Overflow, Quora, Code Comment analysis, Software documentation**

I. Introduction

Recent studies provide evidence that developers consider code comments to be the most important type of documentation for understanding code [1]. Code comments are written using natural language sentences, and their syntax is neither imposed by a programming language's grammar nor checked by its compiler. Consequently, developers follow various conventions in writing code comments [2]. These conventions vary across development environments as developers embed different kinds of information in different environments [3], [4], [5]. This makes it hard to write, evaluate, and maintain the quality of comments (especially for new developers) as the software evolves [6], [7].

To help developers in writing readable, consistent, and maintainable comments, programming language communities, and large organizations, such as Google and Apache Software Foundation provide coding style guidelines that also include comment conventions [8], [9], [10], [11]. However, the availability of multiple syntactic alternatives, the freedom to adopt personalized style guidelines,[1] and the lack of tools for assessing comments, make developers confused about which commenting practice to adopt [6], or how to use a tool to write and verify comments.

To resolve potential confusion, and to learn best commenting practices, developers post questions on various Q&A forums. Stack Overflow (SO) is one of the most popular Q&A forums, enabling developers to ask questions to experts and other developers.[2] Barua *et al.* determined the relative popularity of a topic across all SO posts and discovered the "coding style" topic as the most popular [12]. Similarly, Quora[3] is another widely adopted by developers to discuss software development aspects [13]. However, what specific problems developers report about code comments such as do they face challenges due to multiple writing conventions or development environments, or which commenting conventions experts recommend to them on these sources, is unknown.

Therefore, we analyze commenting practices discussions on SO and Quora, to shed light on these concerns. Particularly, we formulate the following research questions:

1) ***RQ*$_1$*: What* ***high-level topics*** *do developers discuss about code comments?* Our interest is to identify high-level

[1] https://github.com/povilasb/style-guides/blob/master/cpp.rst, accessed on Jun, 2021
[2] https://www.stackoverflow.com
[3] https://www.quora.com

978-1-6654-4897-0/21/$31.00 ©2021 IEEE
DOI 10.1109/SCAM52516.2021.00027

concerns and themes developers discuss about code comments on Q&A platforms.

2) ***RQ***$_2$*: What* ***type of questions*** *do developers ask about code comments?* Our aim is to identify the type of questions developers frequently ask *e.g.*, questions such as how to write comments, or what is the problem in their comments. In addition, we aim to identify which platform they prefer to ask which type of questions.

3) ***RQ***$_3$*: What* ***information needs*** *do developers seek about commenting practices?* We investigate SO and Quora questions in more detail (including body, tags, comments of the question) to identify the challenges and needs related to writing comments in various development environments.

4) ***RQ***$_4$*: What specific commenting conventions are recommended by developers?* We investigate the answers to specific kinds of questions, asking about best practices, to collect commenting conventions suggested by developers.

For each research question, we analyze developer questions at various levels, such as focusing only on the title of the question, the whole question body, or the answers to the question. The rationale behind each level is that future approaches for identifying and automating developers' intent, needs, and recommendations can focus on that specific aspect of comments they want to evaluate and improve. Our manually labeled questions for the research questions RQ_2, RQ_3, and RQ_4 can serve as an initial dataset for building such approaches.

To answer $\mathbf{RQ}_1$ we use a semi-automated approach involving LDA [14], a well-known topic modeling technique used in software engineering, to explore topics from the SO and Quora posts [12], [15], [16], [17]. We then analyze a statistically significant sample set of posts from SO and Quora posts and derive a taxonomy of *types of questions* and *information needs* of developers regarding comments to answer $\mathbf{RQ}_2$ and $\mathbf{RQ}_3$ respectively. To answer $\mathbf{RQ}_4$, we manually analyze the questions asked about best practices on the selected sources and extract various commenting conventions recommended by developers in their answers.

Our results show that developers frequently ask questions on Q&A forums to discuss the best syntactic conventions to write comments, ways to retrieve comments from the code or background information about various comment conventions. Specifically, the questions about how to write or process comments (*implementation strategies*) in a language, tool, or IDE are frequent on SO. On the other hand, questions about background information concerning various conventions or opinions on the best commenting practices are frequently posted on Quora. Our analysis shows that developers are interested in embedding various kinds of information, such as code examples and media (*e.g.*, images), in their code comments but lack the strategies and standards to write them. We also observe a considerable proportion of questions where developers ask about the ways of automating the commenting workflow with documentation tools or IDE features to foster commenting practices and assess them. This shows the increasing need to improve the state of commenting tools by emphasizing better documentation of the supported features and by providing their seamless integration in the development environments.

The contributions of this paper are:

1) an empirically-validated taxonomy of comment-related concerns collected from multiple sources;

2) a first study to investigate the Quora platform for code comments; and

3) a publicly available dataset including all validated data, and steps to reproduce the study in the Replication Package (RP) [18].

Paper structure. In section II we detail the study definition and methodology adopted to answer our research questions. In section III, we present our results and insights. We discuss our findings and their implications in section IV. We recap the threats to validity in section V, summarize related work in section VI, and conclude the paper in section VII.

II. Study Design

The goal of this study is to investigate information needs, practices, and problems developers discuss on various online Q&A platforms about code comments. Figure 1 illustrates the steps followed to answer the research questions.

A. Data collection

SO. To identify the relevant discussions concerning *commenting practices*, we used an approach similar to Aghajani [19]. We selected the initial keywords (Ik) such as *comment*, *convention*, and *doc* to search the SO tags page [19].[4] The search converged to a set of 70 potentially relevant tags, referred to as *initial tags* (It).

Two authors independently examined all the tags, their descriptions and the top ten questions in each tag, and selected the relevant tags. We observed that certain tags are ambiguous due to their usage in a different context, such as the *comments* tag being used in many settings. For example, the tag *comment* (5 710 questions on SO) contains questions about development frameworks (*e.g.*, Django) that provide the feature of attaching comments to a website or other external source (478 questions tagged with "wordpress"), or about the websites where users add comments to the posts (512 questions tagged with "facebook" tag). Therefore, we discarded posts where co-appearing tags were *wordpress* and *facebook*, or *django-comment*. The resulting set of tags (It) is therefore reduced to 55 tags out of 70 tags. The list of selected 55 tags is available in the RP.[5]

[4] https://stackoverflow.com/tags verified on Jun 2021

[5] File "RP/Tags-topics.md" in the Replication package

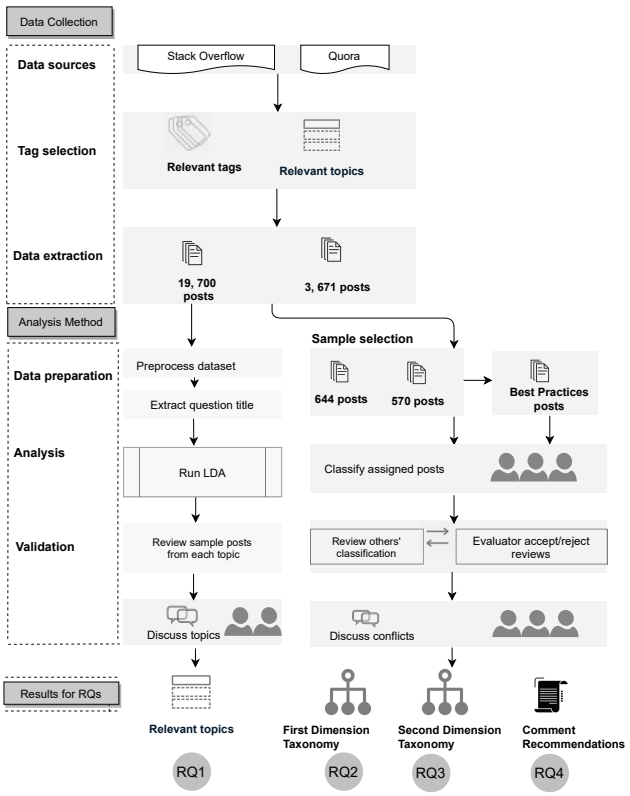

Fig. 1. Research Approach to answer research questions

We extracted all questions tagged with at least one tag from the It set, resulting in 19 705 non-duplicate questions. For each question, we extracted various metadata fields such as ID, title, body, tags, creation date, and view count from SO using the *Stack Exchange Data Explorer* interface.[6] The interface facilitates the users to query all Stack Exchange sites in an SQL-like Query language.[7]

Quora. Extracting data from Quora was non-trivial due to the lack of publicly available datasets and services to access the data, its restrictive scraping policies [20], and the absence of a public API to access the data. Thus, to extract its data, we implemented a web scraper in Python using *selenium* to automate browsing, and *BeautifulSoup* to parse the HTML.[8] On Quora, the notion of topics is the same as tags on SO, so a question in Quora can be tagged with topics similar to SO tags. Unlike the SO tag page, Quora provides neither an index page listing all its topics, nor a list of similar topics on a topic page. We therefore used the relevant SO tags as initial Quora topics, searched for them on the Quora topic search interface, and obtained 29 topics, such as *Code Comments*, *Source Code*, *Coding Style*. The list of all topics and their mapping to SO tags is provided in the RP.[9] We scraped all questions with their meta-data such as URL, title, body, and topics of each question from the identified topics, resulting in 3 671 questions in total.

B. Analysis Method

Automated analysis from LDA ($\mathbf{RQ}_1$). LDA infers latent discussion topics to describe text-based documents. Each document can contain several topics, and each topic can span several documents, thus making it possible for the LDA model to discover ideas and themes in a corpus. We applied LDA on the SO dataset but excluded the Quora dataset as it contains a high number of irrelevant posts (nearly 80%) based on the manually analyzed statistically significant sample set shown in Table I. Additionally, as LDA uses the word frequencies and co-occurrence frequencies across documents to build a topic model of related words, having a high number of irrelevant posts can impact the model quality. Since our objective is to discover the high-level concerns developers have, we extract only titles of the SO questions, as the title summarizes the main concern while the body of the question adds non-relevant information, such as details of development environment, what the developer has tried, or sources already referred to.

To achieve reliable high-level topics from LDA, we performed the following data-preprocessing steps on the question titles: removal of HTML tags, code elements, punctuation and stop words (using the Snowball stop word list[10]), and applied Snowball stemming[21]. We used the data management tool, Makar, to prepare the data for LDA [22]. We provide the concrete steps Makar performed to preprocess the data in the *Reproducibility* section in the online appendix [18]. The preprocessed *title* field of the questions served as the input documents for LDA. We used the *Topic Modeling Tool* [23], a GUI for MALLET [24] that uses a Gibbs sampling algorithm, and facilitates extending the results with meta-data. We provide the input data[11] used for the MALLET tool and the output[12] achieved in the RP.

LDA requires optimal values for the k, α, and β parameters to be chosen, which depends on the type of data under analysis, but this represents an open challenge in software engineering tasks. Wallach *et al.* pointed out that choosing a smaller k may not separate topics precisely, whereas a larger k does not significantly vary the quality of the generated topics [25]. Therefore, to extract distinct topics that are both broad and high-level, we experimented with several values of k ranging from 5 to 25, as suggested by Linares-Vsquez *et al.* [26]. We assessed the optimal value of k by analyzing the topic distribution, coherence value (large negative values indicate that words do not co-occur together often)[13] [27], and perplexity score (a low value means the model correctly predicts unseen words) [28] for each value of k from the given

[6] https://data.stackexchange.com/
[7] File "RP/Stack-exchange-query.md" in the Replication package
[8] [https://pypi.org/project/beautifulsoup4/], [https://www.selenium.dev/]
[9] File "RP/Tags-topics.md" in the Replication package
[10] http://snowball.tartarus.org/algorithms/english/stop.txt
[11] Folder "RP/RQ1/LDA_input" in the Replication package
[12] Folder "RP/RQ1/LDA_output" in the Replication package
[13] http://mallet.cs.umass.edu/diagnostics.php

range [29]. This process suggested $k = [10]$ as the most promising value for our data (with the lowest perplexity of -6.9 and high coherence score of -662.1) as fewer redundant topics were selected with these values.

In the next iterations, we optimized the hyperparameters α and β by using the best average probability of assigning a dominant topic to a question, inspired by the existing studies [30]. We selected the initial values of hyperparameters $\alpha = \frac{50}{k}\beta = 0.01$ using the de facto standard heuristics [31] but allowed these values to be optimized by having some topics be more prominent than others. We ran the model optimizing after every ten iterations in total 1000 iterations. Thus, we concluded that the best hyperparameter configuration for our study is $k = 10, \alpha = 5, \beta = 0.01$. As LDA does not assign meaningful names to the topics, we manually inspected a sample of 15 top-ranked questions under each topic to assign topic names.

TABLE I
DATA EXTRACTED FROM SO AND QUORA SOURCES

Source	Extracted posts	Manually analyzed	Relevant posts
SO	19 700	644	416
Quora	3 671	565	118

Taxonomy Study (RQ_2, RQ_3). After extracting all posts tagged with the relevant tags for SO, we analyzed a statistically significant sample from all (19 700) posts, reaching a confidence level of 99% and an error margin of 5%. The resulting sample set contains 644 posts. We selected the sample posts using a random sampling approach without replacement to reach 644 posts. Similarly, for Quora, we selected 565 posts for our manual analysis, as shown in Table I.

Classification: We classified the sample posts into a *two-dimensional taxonomy* mapping concepts of the selected questions. The first dimension (*question types*) aims to answer RQ_2 while the second dimension (*information needs*) answers RQ_3. The first dimension, inspired from an earlier SO study [32], defines the categories concerning the kind of question, *e.g.*, if a developer is asking how to do something related to comments, what is the problem with their comments, or why comments are written in a particular way. We renamed their categories [32] to fit our context, *e.g.*, their 'What' type of question renamed to 'Implementation problems.' To classify the selected sample questions in the first dimension categories as shown in Table II, we used closed card sorting technique.

The second dimension outlines more finely-grained categories about the types of information needs developers seek [33], *e.g.*, development environment-related needs (*e.g.*, comments in programming languages, tools, IDEs), or about comments in general. The majority of these categories are built based on the software documentation work by Aghajani *et al.* [19], and the questions are classified into these categories using the hybrid card sorting technique [34]. In the development environment-related needs, we identified if a question talks about *IDE & Editors* (*e.g.*, Intellij, Eclipse), *Programming languages* (Java, Python), or *Documentation tools* (Javadoc, Doxygen). The further sub-levels of the taxonomy focus on the type of information a questioner is seeking in each development environment, such as asking about the syntax to add a comment or specific information in the comment in Javadoc [33]. For instance, the question *"How to reference an indexer member of a class in C# comments"* [35] is about the C# language, and asking about the syntax to refer to a member in the class comment, thus gets classified into the three levels as *Programming languages—Syntax & format—Class comment* according to the taxonomy shown in Figure 6.

Execution and Validation: A Ph.D. candidate, a master's student, and a faculty member, each having more than three years of programming experience, participated in the evaluation of the study. The sample set of questions was divided into an equal subset of questions and selected random questions for each subset to ensure that each evaluator gets a chance to look at all types of questions. We followed a three-iteration-based approach to categorize the questions. In the first iteration, we classified the posts into first and second-dimension categories. In the second iteration, each evaluator (as a reviewer) reviewed the classified questions of other evaluators and marked their agreement or disagreement with the classification. In the third iteration, the evaluator agreed or disagreed with the decision and changes proposed by the reviewers. In case of disagreements, another reviewer who had not yet looked at the classification reviewed the classification and gave his/her decision. Finally, if all evaluators disagreed, we chose the category based on the majority voting mechanism. This way, it was possible to ensure that each classification is reviewed by at least one other evaluator. In case of questions belonging to more than one category, we reviewed the other details of questions, such as tags and comments of the questions and chose the most appropriate one. We finalized the categories and their names based on the majority voting mechanism.

Based on the classification and validation approach described above, all three authors (evaluators) evaluated first the relevance of all their assigned questions, and reviewed the cases of irrelevant questions marked by other evaluators. The third author reviewed and resolved their disagreement cases using a majority voting mechanism (cohen's k = 0.80). As a result, 416 questions of SO and 118 questions of Quora were considered relevant to our study, as shown in Table I. The remaining questions, marked as irrelevant, were manually inspected and no new relevant topic was identified.

Recommended comment conventions (RQ_4). Given the unstructured or semi-structured nature of comments, and varying standards to write comments, various organizations and language communities present numerous commenting guidelines to support consistency and readability of the comments. For instance, the convention "*Use 3rd person (descriptive) not 2nd person* in writing comments" is given in the Java Oracle style

TABLE II
FIRST DIMENSION CATEGORIES

Category	Description	Question keywords to identify with an example
Implementation Strategies	The questioner is not aware of ways to write or process comments. They often ask questions about integrating different information in their comment, using features of various tools.	"How to", *e.g.*, *How to use @value tag in javadoc?*
Implementation Problems	The questioner tried writing or processing the code comment but was unsuccessful.	the question "What is the problem?", *e.g.*, *Doxygen \command does not work, but @command does?*
Error	The questioner posted the error, exceptions or crashes while writing or generating comments, or any warning produced by the documentation tool.	contain an error message from the exceptions or stack trace
Limitation & Possibilities	The questioner is seeking more information about limitations of a comment related approach, tool, or IDE, and various possibilities to customize the comment.	the question "is it possible or allowed", *e.g.*, *Is there a key binding for block comments in Xcode4?*
Background Information	The questioner is looking for background details on the behavior of comments in a programming languages, a tool, or a framework.	the question "why something", *e.g.*, *Why in interpreted languages the # usually introduces a comment?*
Best practice	The questioner is interested to know the best practice, guidelines or general advice to tackle a comment-related problem or convention.	the question "is there a better way to", *e.g.*, *What is the proper way to reference a user interaction in Android comments?*
Opinion	The questioner is interested to know the judgment of other users for a comment convention.	the question "what do you think", *e.g.*, *Are comments in code a good or bad thing?*

guide. However, not all of these conventions are recommended by developers in real-time and some conventions are even discouraged, depending on the development environment. Additionally, developers assumed some conventions were feasible *e.g.*, overriding docstrings of a parent class in its subclasses, but other developers pointed out them as a limitation of current documentation tools or environment. We attempted to collect such comment conventions recommended by developers in their answers on SO and Quora. From the classified questions in RQ_2, we chose the questions categorized in the *Best Practice* category according to Table II. Based on the accepted answers of these questions, we identified *recommendation* or *limitation* of various comment conventions. In case a question has no accepted answer, we referred to the top-voted answer.

III. RESULTS

A. High-Level Topics Discussed about Comments (RQ_1)

Table III shows the 10 topics generated from the LDA analysis, where the column *Relevance* denotes if a topic is relevant (R), or irrelevant (IR) and *Topic name* is the assigned topic label. The column *Topic words* shows the words generated by LDA, sorted in the order of their likelihood of relevance to the topic.

In the *Syntax & Format* topic, developers mainly ask about the syntax of adding comments, removing comments, parsing comments, or regex to retrieve comments from code. Occasionally, the questions are about extracting a particular type of information from comments to provide customized information to their clients, such as obtaining descriptions, or to-do comments from code comments. Depending on a programming language or a tool, strategies to add information in the comments vary, such as adding a description in XML comments [*SO:9594322*], in the R environment [*SO:45917501*], or in the Ruby environment [*SO:37612124*]. This confirms the relevance of recent research efforts on identifying the comment information types in various programming languages [3], [4], [36]. *IDEs & Editors* groups questions about commenting features provided in various IDEs to add or remove comments in the code, or setting up documentation tools to write comments. *R Documentation* groups questions about documentation features provided in the R language, such as creating various formats of documents including Markdown, Latex, PDF, or HTML documentation. 'R Markdown', a documentation format available in knitr package, provides these features in R. In the *Code convention* topic groups the questions about best practices, such as printing the docstrings of all functions of an imported module, or conventions to check types in Javascript. In this topic, developers also ask the reasons behind various code conventions such as the reason behind having only one class containing the main method in Java, or using particular symbols for comments. In the *Documentation generation* topic, developers inquire about problems in generating project documentation or HTML documentation from comments automatically using various documentation tools such as Sphinx, Doxygen. Apart from code comments, software projects support other forms of documentation, such as wikis, user manuals, API documentation, or design documentation. As the project documentation is divided into various components, developers post questions about locating them in the topic *Seeking documentation*. Additionally, developers also showed interest in learning various programming languages and thus seeking documentation to learn them.

Finding 4.1.1 The most relevant topics discussed by developers about commenting practices, and identified by LDA are related to *Syntax & format*, *Documentation generation*, *IDEs & Editors*, *R documentation*, *Code conventions*, and *Seeking documentation*, indicating developers interest in automating their code documentation.

B. Types of Questions Discussed (RQ_2)

To get insights about which types of questions developers ask about comments and where they ask the specific types of questions more often, we categorized the sampled set of questions from SO and Quora according to the first dimension (*question type*) shown in Table II. Figure 3 shows such categories on the x-axis with respect to both sources, and the y-axis indicates the percentage of questions belonging to a category out of the total question of a source. The figure highlights *implementation strategies* (how-to) to be the most

TABLE III
TOPICS GENERATED BY LDA WITH ASSIGNED TOPIC NAME AND TEN MOST IMPORTANT TOPIC KEYWORDS

Relevance	Topic name	Topic words
R	Syntax & format	line code file c python doxygen block php html text remov string tag javascript style add regex command script singl
R	IDEs & Editors	javadoc generat studio eclips visual file xml java project code class c android sourc maven tag doc intellij show netbean
R	R documentation	r rmarkdown markdown tabl output pdf html file knitr code render chunk text packag chang latex error knit plot add
R	Code conventions	function class method jsdoc type doxygen paramet c object variabl return python phpdoc javadoc name refer properti convent valu docstr
IR	Development frameworks for thread commenting	api doc rest rail generat spring rubi test net swagger web rspec where asp creat find develop googl rdoc what
IR	Open source software	code sourc what open can app where whi get find anyon websit develop program if android mean softwar doe someon
R	Documentation generation	sphinx file doxygen generat python html link doc page includ modul make creat build custom autodoc rst restructuredtext imag output
IR	Thread comments in websites	facebook post wordpress page get php whi plugin user box like show youtub section display form system repli delet add
IR	Naming conventions & data types	convent name what java python develop c whi sql tabl valu case mysql string data x program code variabl column
R	Seeking documentation & learning language	what code software best program way write good language develop standard requir tool project c practic need learn are which

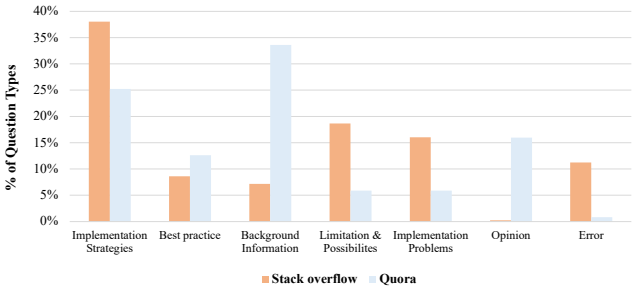

Fig. 2. First dimension categories found on SO and Quora

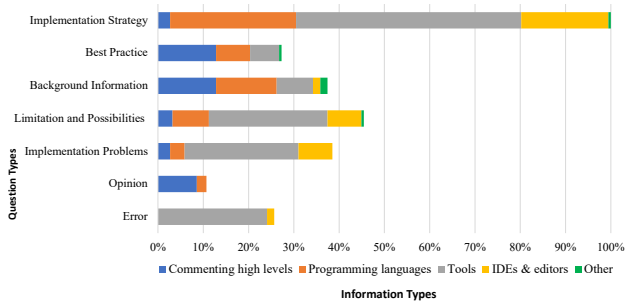

Fig. 3. Distribution of first dimension and second dimension categories

frequent category on SO, confirming prior study results [37], [38], [15], [12]. Differently from previous studies, we found *best practice* and *background information* questions to arise more frequently than *implementation problems* questions.

We also observed that different types of questions are prevalent on the investigated platforms, as highlighted by Figure 2. The figure shows that developers ask *implementation strategies* questions and *implementation problems* questions more on SO compared to Quora. Despite Quora being an opinion-based Q&A site, we also observed questions about *best practice* and *background information* about reasons behind various implementation and symbols used in comments. This shows how developers rely on Quora to gather knowledge behind numerous conventions and features provided by the development environment. Such types of questions are also found on SO but to a lesser extent. For instance, we observed the question: *What are these tags @ivar @param and @type in python docstring* [*SO:379346*] on SO. Based on the thousands of views count of the post, we can say that the question has attracted the attention of many developers. Uddin *et al.* gathered developers' perceptions about APIs in SO by mining opinions automatically [39]. Our study provides the evidence to include Quora as another source to validate their approach and mine developer's opinions.

Finding 4.2.1 Different kinds of questions are prevalent on SO and Quora *e.g.*, *implementation strategies* and *implementation problems* questions are more common on SO whereas *best practice* and *background information* questions apart from opinion-based questions are more prevalent on Quora. This suggests that Quora can be a useful resource to understand how developers perceive certain development aspects, while SO is useful to understand what technical challenges they face during its development.x

C. Developer Information Needs (***RQ***$_3$)

We analyzed the questions from two different perspectives. The first-dimension categories (*question types*) (Y-axis in Figure 3) show the types of question (*e.g.*, implement strategies, problems) developers ask, whereas the second-dimension categories (X-axis in Figure 3) highlight the kinds of problems they face with the code comments in the development environment. For example, Figure 3 shows how in the second dimension analysis the most frequent category *implementation strategies* contains questions about how to do something (comment-related) in a development environment, be it specific to a programming language, a documentation tool, or to the IDE itself. On the other hand, developers discuss the

possible features of the documentation tools and IDEs in the *limitation & possibilities* category. This category highlights the developer's struggle in locating the feature details from the documentation of such tools and showcases the vital need for improving this aspect. However, which specific features and syntaxes of comments developers seek in the development environment is essential information to progress in this direction. Therefore, we first separated general questions about comment conventions to the *commenting high levels* category, and moved other development environment related questions to the *programming languages*, *tools*, and *IDEs & editors* categories (first-level categories shown in Figure 6). We then added sub-categories such as *Syntax & format*, *Asking for feature*, *Change comment template etc.* under each first-level category to highlight the specific need related to it, as shown in Figure 6 and explained in Table III [18]. In the next paragraphs, we explain a few such subcategories.

Finding 4.2.2 One-third of the developer commenting practices questions are about their specific development environment. The top two frequent questions concern the categories *Syntax & format* and *Asking for feature* indicating developers' interest in improving their comment quality. The rest focus on setting up or using documentation tools in IDEs to generate comments automatically.

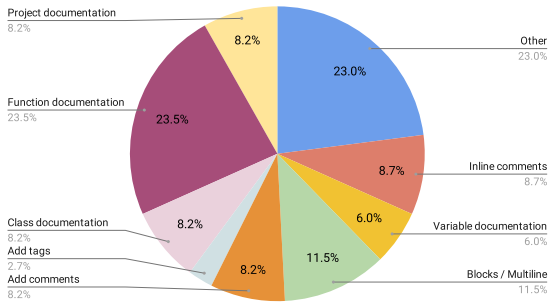

Fig. 4. Comment's syntax & format discussions

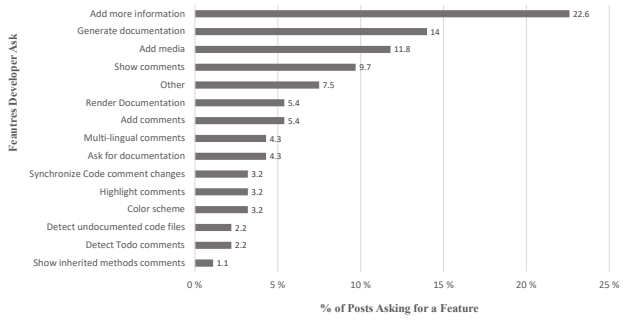

Fig. 5. Developers asking for features

In *Syntax & format*, shown in Figure 4, developers discuss syntax to add various kinds of comments to the code, such as function comments, class comments, block comments, and different tags. Specifically, the syntax of writing function comments is asked more frequently (23% of questions) than other types of comments, showing the importance and efforts of API documentation. In this analysis we found 23% of the questions marked as *Other* are either about the syntax of writing comments in a programming language or a tool without stating the kind of comment (class/function/block), or concerning the intent of syntax conventions. Such background questions are more often posted on Quora compared to SO.

Asking for feature is another frequent information developers seek on SO to locate various features provided by the documentation tools. We rarely found such questions on Quora. Aghajani *et al.* reported a similar category as *Support/Expectations* covering developer needs that are not satisfied by the documentation tools in their work [19]. In our study context, we reported such inquiries, as shown in Figure 5, and in the category *Asking for feature* under all development environment categories in Figure 6.

Figure 5 shows that developers frequently need to add different kinds of information to the code comments, such as code examples: *How to add code description that are not comments?* [*SO:45510056*], performance-related: *Javadoc tag for performance considerations* [*SO:39278635*], and media [*SO:43556442*]. Additionally, developers ask about features to add comments automatically, detect various information from the comments, or synchronize the comments with code changes [*SO:23493932*]. These questions show the worthiness of devoting research efforts to the direction of identifying information types from comments, detecting inconsistent comments, and assessing and generating comments automatically to improve code comments [3], [40]. We separated the feature-related questions (different features of the tools and IDEs) into two categories, *Using feature* and *Asking for feature*, based on the user awareness. In the former category, the user is aware of the existence of a feature in the environment but finds problems in using it, as shown in Listing 1. In the latter category, users inquire about the existence of a feature, or try to locate it, as shown in Listing 2.

```
How to use @value tag in javadoc?
```

Listing 1. Using @value feature in Javadoc

```
How can I show pictures of keyboard keys in-
    line with text with Sphinx?
```

Listing 2. Asking for a feature to add inline images

Finding 4.2.3 The *implementation strategies* category questions are the most frequently viewed questions on SO and *limitation & possibilities* (is it possible) questions are the second most viewed questions based on the view count of questions. On the other hand, based on the answer count of questions, *best practice* questions trigger the most discussions along with *implementation strategies*.

In addition to the above categories, we observed that SO encourages developers, especially novice developers, to ask questions about the basics of various topics [41], grouped into the *commenting high levels* category, and shown in Figure 6. This detailed taxonomy of the second dimension is reported in

the Figure 6 and Table III in the online appendix [18]. Figure 6 reports all levels of the second dimension according to the source. For instance, the questions about setting up tools (*Tool setup*), or asking for various features (*Asking for features*) under *IDE & editors* are not found on Quora. Similarly, the majority of the questions about documentation tools (*Tools*) are asked on SO whereas the general questions about comments (*Commenting high levels*) are often on Quora.

Finding 4.2.4 Developers often ask about the syntax to write function (method) comments compared to other kinds of comments (class, package), showing the trend of increasing effort towards API documentation. Another frequently asked question on SO concerns the conventions to add different kinds of information to code comments, such as code examples, media, or custom tags, indicating developers' interest in embedding various information in comments.

Finding 4.2.5 Apart from questions related to comment syntax and features, developers ask about adopting commenting styles from other programming languages, modifying comment templates, understanding comments, and processing comments for various purposes.

D. Recommended Comment Convention ($\boldsymbol{RQ}_4$)

There are various syntactic and semantic commenting guidelines mentioned in the style guides, and developers are often confronted with several conventions, or are unable to find any for a specific purpose. We collect various comment conventions recommended by developers in their answers on SO and Quora in Table IV. For example, a developer asks *Should .net comments start with a capital letter and end with a period?*[*SO:2909241*], concerning grammar rules in the comments. The accepted answer affirms the convention and describes how it helps to improve readability. We, therefore, constructed the recommendation *[.net] long inline comments should start with a capital letter and end with a period*. In some answers, developers describe it as a limitation, we included *Limitation* for such answers. For each recommendation, we indicate whether it is specific to a programming language, a tool, an IDE, or is instead a general recommendation, using tags such as "[Java], [Doxygen], [visual studio],[general]" respectively. It is important to note that we did not verify how widely the recommendations are adopted in the commenting style guidelines or projects, or how well they are supported by current documentation checker tools (or style checkers). This is a future direction for this work. On the positive side, it represents an initial starting point to collect various comment conventions confirmed by developers. We argue that it can also help researchers in conducting the studies to assess the relative importance of comment conventions or help tool developers in deciding which recommendation they should include in their tools to address frequent concerns of developers.

TABLE IV
CODE COMMENT CONVENTIONS RECOMMENDED BY DEVELOPERS

Topic	Recommendation
Grammar	[.net] long inline comments should start with a capital letter and end with a period.
	[.net] long inline comments should be written as a complete English sentence (with subject, verb, object).
	[general] check your coding style guidelines to verify how to write plural objects in the comments, for example, Things(s) or Things.
	[general] Do not mark the code section with an inline comment to highlight the modified code section, version control system keep track of code changes.
	[python] use backslash escape whitespace to use punctuations like apostrophe symbol in docstring.
	[python] Use 'truthy' and 'falsy' words to denote boolean values 'True' and 'False' respectively.
	[general] Do not write filler words such as 'please' and 'thank you', nor swearing words in the comments.
	[general] Remove TODO comments when you finish the task.
Language	[general] Comments should explain why and not how.
	[general] Use correct notation to write block or multiline comments.
	[general] Position your inline comments (about variable declaration) above the variable declaration to remain consistent with method comment conventions.
	[general] Do not write nested comments in the code.
	[general] Use different tags to categorize the information in the comments.
	[general] Do not use multiple single line comments instead of multi-line comments.
	[general] Do not document file specifications in the code comments rather document them in the design specs.
	[general] Use a consistent style such as *'variable'* or *<variable>* to differentiate the code variable names in the inline comments.
	[Java] Implementation notes about the class should be mentioned before the class definition rather than inside the class.
	[Java] To denote a method (*someMethod() of the class ClassA*) in the comments, use the template *the <someMethod> method from the <ClassA> class* instead of *ClassA.someMethod()*.
	[.net] Document 'this' parameter of an extension method by describing the need of 'this' object and its value.
	[javascript] *Limitation:* Currently, there is no existing standard to document AJAX calls of javascript in PhpDoc style comments.
	[php] Use '->' symbol to reference instance/object method rather than '::' in the method comments.
	[sql] Use the same documentation style for SQL objects as you are using for other code.
	[groovy] Limitation: there is no standard way to document properties of a dynamic map in Javadoc like JSDoc' @typedef.
Tool	[PhpDoc,JsDoc] Do not put implementation details of a public API in the API documentation comments, rather put them in inline comments inside the method.
	[JSDoc] Mention the class name in description to denote the instance of the class.
	[ghostDoc] Create your default comment template using c# snippets.
	[JavaDoc] *Limitation:* Currently it is not possible to generate documentation of an API in multi-languages (in addition to English) with the same source code.
	[JavaDoc] *Limitation:* The tag @value support fields having literal values. JavaDoc and IntelliJ IDEA do not support fetching value from an external file using @value tag in Javadocs.
	[Javadoc] Write annotations after the method javadoc, before the method definition.
	[Doxygen] Use @copydoc tag to reuse the documentation from other entities.
	[Doxygen] *Limitation:* Currently it is not possible to generate documentation of an API for different readers such as dev and users.
	[Doxygen] Use @verbatim / @endverbatim to document console input and output.
	[Roxygen] Limitation: Not possible to override docstrings so the parent docstring is used when inheriting a class.
	[PhpDoc] *Limitation:* Currently it is not supported to document array details in the return type of a method.
	[PhpDoc] *Limitation:* Currently, using @value tag or any similar tag to refer to the value of a field is not supported in PhpDoc, so developers should use @var tag instead.
	[Phpdoc] Use class aliases in import statement to write short name in docblock.
	[Sphinx] Limitation: It can't create sections for each class. Add yourself the sections in the .rst file.

IV. DISCUSSION AND IMPLICATION

On Writing Comments. Although, various coding style guidelines provide conventions to write comments, our results showed that SO developers seek help in writing correct syntax of various comment types (class/function/package comments highlighted in Figure 4), in adding specific information in comments, or formatting comments. Typical types of questions are *What is the preferred way of notating methods in comments?* [*SO:982307*], and *Indentation of inline comments in Python* [*SO:56076686*], or *indentation of commented code* [*SO:19275316*]. This indicates the need of improving the commenting guideline and assuring their findability to developers. Tomasottir *et al.* showed in their interview study that developers use linters to maintain code consistency and to learn about the programming language [42]. By configuring linters early in a project, developers can use them similarly to learn the correct syntax to write and format comments

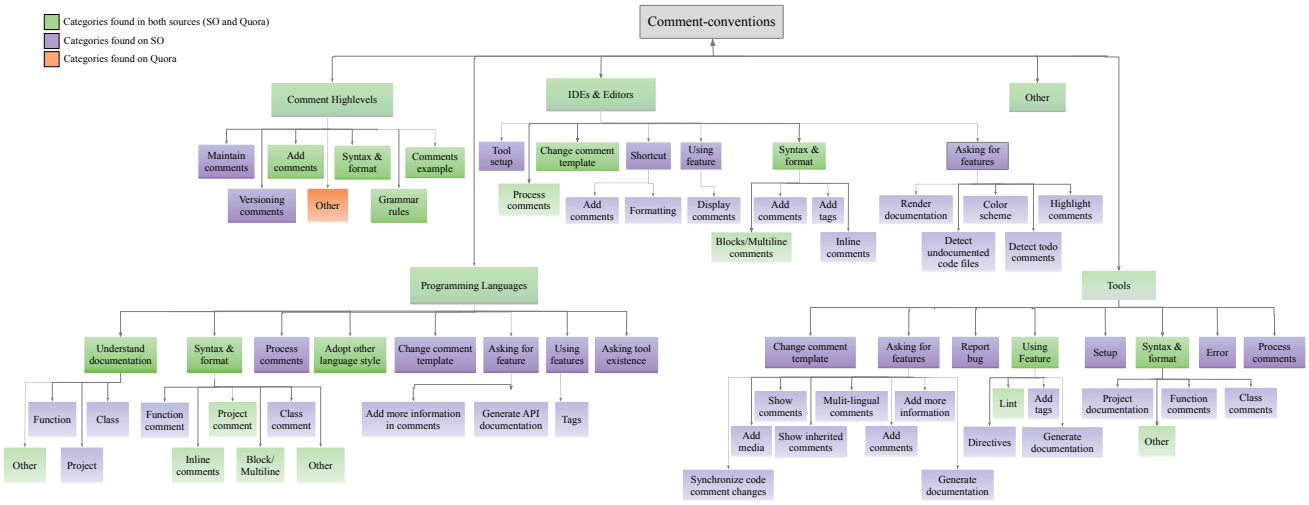

Fig. 6. Taxonomy of second dimension on SO and Quora

according to a particular style guideline. However, due to their support to multiple languages, assisting developers in language-specific conventions, or customizing comments to add more information would still require further effort.

On Coding Style Guidelines. Organizing the information in the comments is another concern highlighted in the study, for example, *how to differentiate the variables within a comment* [*SO:2989522*] [*SO:47089022*], *where to put class implementation details* (in the class comment or in inline comments) [*SO:35957906*], and *which tag to use for a particular type of information* [*SO:21823716*]. We also found developer concerns regarding grammar rules and word usage in all the sources we analyzed (SO [*SO:2909241*], and Quora [43]). Although various style guidelines propose comment conventions, there are still many aspects of comments for which either the conventions are not proposed or developers are unable to locate them. Developers commonly ask questions, such as *Any coding/commenting standards you use when modifying code?* [*SO:779025*] on SO and *Why is there no standard for coding style in GNU R?* on Quora. There is therefore a need to cover detailed aspects of comments in the coding style guidelines to help developers write high-quality comments.

On the Impact of Comment Conventions. Various commenting conventions are presented in the style guides to support consistent and readable comments. However, extracting these conventions automatically from style guidelines and customizing them according to the project requirements is still a challenge and not explored much. Additionally, which of these conventions play a more important role for comment comprehension and which ones do not, is not yet explored. Binkley *et al.* evaluated the impact of identifier conventions on code comprehension, but the conventions were limited to identifiers [44]. Smit *et al.* identified the relative importance of 71 code conventions, but the majority of the comment conventions were limited to detecting missing documentation comments [45]. Therefore, assessing the impact and importance of comment conventions depending on a specific domain and project, and on various development tasks appears to be another potential direction.

Tools to Assess Comment Quality. Our results show that developers are interested in various automated strategies, such as automatic generation of comments, detection of bad comments, identification of information embedded in comments, and the quality assessment of comments, lack tools that can be integrated into their IDE, especially to verify the comment style automatically [*SO:14384136*]. However, a limited set of documentation tools support comment quality assessment or adherence of comment to the commenting conventions. For example, current style checker tools, such as Checkstyle, RuboCop and pydocstyle provide support for formatting conventions but lack support for comprehensive checks for grammar rules and content.[14] There is a need to survey current automated style checker tools. Additionally, some languages with advanced style checkers don't support comment checkers at all, such as OCLint for Objective-C, and Ktlint for Kotlin, Smalltalk.[15] We found instances of developers asking about the existence of such tools [*SO:8834991*] in Figure 5 and *Asking tool existence* in Table III in the online appendix [18]. Therefore, more tool support is needed to help developers in verifying the high-quality of comments.

V. Threats to Validity

Threats to construct validity concern the relationship between theory and experimentation. In our study, they mainly relate to potential imprecision in our measurements. To mitigate potential bias in the selection of developer discussions on SO, we relied on SO tags to perform initial filtering. However, it is possible that this tag-based filtering approach misses some relevant posts concerning comment convention practices

[14] [https://checkstyle.org/checks.html], [https://rubocop.org/], [http://www.pydocstyle.org/]
[15] [http://oclint.org/], [https://github.com/pinterest/ktlint]

and topics. We therefore investigated the co-appearing tags to find similar relevant tags. Aghajani *et al.* studied software documentation-related posts, including code comments on SO and other sources [19]. We extracted their documentation-related tags from the given replication package and compared them to our tags (It) to verify if we missed any. On Quora, we mapped the selected SO tags as keywords and searched these keywords on Quora search interface. To avoid eventual biased in this manual process, we also adopted LDA, to investigate high-level topics emerging in the SO and Quora posts. Thus, a mix of qualitative and quantitative analysis was performed to minimize potential bias in our investigation, providing insights and direction into the automated extraction of relevant topics.

Threats to internal validity concern confounding factors, internal to the study, that can affect its results. In our study, they mainly affect the protocol used to build the taxonomy, which could directly or indirectly influence our results. To limit this threat, we used different strategies to avoid any subjectivity in our results. Specifically, all posts were validated by at least two reviewers and, in case of disagreement, a third reviewer participated in the discussion to reach a consensus. Thus, for the definition of the taxonomy, we applied multiple iterations, involving different authors of this work.

Threats to conclusion validity concern the relationship between theory and outcome. In our study, they mainly relate to the extent to which the produced taxonomy can be considered exhaustive. To limit this threat, we focused on more than one source of information (SO and Quora), so that the resulting taxonomy has a higher likelihood to be composed of an exhaustive list of elements (*i.e.*, comment convention topics).

Threats to external validity concern the generalizability of our findings. These are mainly due to the choice of SO and Quora as the main sources. SO and Quora are widely used for development discussions to date, although specific forums, such as DZone and Reddit could be considered for future works.[16] Moreover, besides all written sources of information, we are aware that there is still a portion of the developer communication taking place about these topics that are not traceable. Thus, further studies are needed to verify the generalizability of our findings.

VI. Related Work

Studying developer activities related to various development tasks from various sources can guide the development of tools that help developers to find the desired information more easily. Therefore, researchers have recently focused on leveraging the useful content of these sources *i.e.*, Git, CVS [21], archived communications online forums and CQA (Community Question Answer) sites [38], [12], [30], [16], [13], [19] to comprehend developers information needs. SO is one of the more popular platforms that researchers have studied to capture developers questions about trends and technologies [38], security-related issues [16], and documentation issues *etc.* [19]. Recently researchers have started investigating Quora to get more insight into developer communities [13], *e.g.*, finding and predicting popularity of the topics [46], [13], finding answerability of the questions [47], detecting experts on specific topics [20], [48], [49], or analyzing anonymous answers [50] Our study is first to investigate this platform for code comments.

Research shows that interesting insights can be obtained from combining these sources [51], [19]. Aghajani *et al.* studied documentation issues on SO, Github, and mailing lists [19]. They reported a taxonomy of documentation issues developers face. However, they do not focus on the style issues of the code comments. Our study focuses on the all aspects of code comments *i.e.*, the content and style aspect of the code comments. In a previous study, Barua *et al.* found coding style/practice among the top share on SO [12]. They considered the topic among common English language topics instead of a technical category due to usage of generic words in this topic. As their focus was on technical categories, they did not explore the coding style questions further. Our study complements their work by exploring the specific aspects of coding style, focusing on comment conventions.

VII. Conclusions

In this study, we investigated commenting practices discussions occurring in SO, and Quora. We first performed automated analysis (LDA) on extracted discussions and then complemented it with a more in-depth manual analysis on the selected sample set. From the manual analysis, we derived a two-dimensional taxonomy. The first dimension of the taxonomy focuses on the question types, while the second dimension focuses on five types of first-level concerns and 20 types of second-level concerns developers express. We qualitatively discussed our insights, and presented implications for developers, researchers and tool designers to satisfy developer information needs regarding commenting practices. We provide the data used in our study, including the validated data and the detailed taxonomy, in the replication package [18]. In the future, we plan (i) to verify the completeness and relevance of gathered comment conventions, (ii) to survey practitioners and tool designers to learn which rules are more important than others, and (iii) to explore ways to improve and assess tool support in this direction. This investigation will help us to see which comment convention affects them most, and during which specific development activity.

Acknowledgments

We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project "Agile Software Assistance" (SNSF project No. 200020-181973, Feb. 1, 2019 - April 30, 2022).

[16] [https://dzone.com/articles/my-commentary-on-code-comments], [https://www.reddit.com]

REFERENCES

[1] S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira, "A study of the documentation essential to software maintenance," in *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information*, ser. SIGDOC '05. New York, NY, USA: ACM, 2005, pp. 68–75.

[2] Y. Padioleau, L. Tan, and Y. Zhou, "Listening to programmers — taxonomies and characteristics of comments in operating system code," in *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 2009, pp. 331–341.

[3] L. Pascarella and A. Bacchelli, "Classifying code comments in Java open-source software systems," in *Proceedings of the 14th International Conference on Mining Software Repositories*, ser. MSR '17. IEEE Press, 2017, pp. 227–237. [Online]. Available: https://doi.org/10.1109/MSR.2017.63

[4] J. Zhang, L. Xu, and Y. Li, "Classifying python code comments based on supervised learning," in *Web Information Systems and Applications - 15th International Conference, WISA 2018, Taiyuan, China, September 14-15, 2018, Proceedings*, ser. Lecture Notes in Computer Science, X. Meng, R. Li, K. Wang, B. Niu, X. Wang, and G. Zhao, Eds., vol. 11242. Springer, 2018, pp. 39–47. [Online]. Available: https://doi.org/10.1007/978-3-030-02934-0_4

[5] P. Rani, S. Panichella, M. Leuenberger, M. Ghafari, and O. Nierstrasz, "What do class comments tell us? An investigation of comment evolution and practices in Pharo Smalltalk," *arXiv preprint arXiv:2005.11583*, 2020, to be Published in Empirical Software Engineering.

[6] M. Allamanis, E. T. Barr, C. Bird, and C. Sutton, "Learning natural coding conventions," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 281–293. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635883

[7] B. W. Kernighan and R. Pike, *The Practice of Programming (Addison-Wesley Professional Computing Series)*, 1st ed. Addison-Wesley, Feb. 1999. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/020161586X

[8] "Oracle documentation guidelines," 2020. [Online]. Available: https://www.oracle.com/technetwork/java/javase/documentation

[9] W. S. Jr. and E. White, *The Elements of Style*, 4th ed. Allyn and Bacon, 2000.

[10] "Google style guidelines," 2020, verified on 10 Jan 2021. [Online]. Available: https://google.github.io/styleguide/

[11] "Apache spark code style guide." [Online]. Available: https://spark.apache.org/contributing.html

[12] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in Stack Overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014. [Online]. Available: https://doi.org/10.1007/s10664-012-9231-y

[13] J. Krüger, "Are you talking about software product lines? An analysis of developer communities," in *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems*, 2019, pp. 1–9.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[15] S. Wang, D. Lo, and L. Jiang, "An empirical study on developer interactions in Stack Overflow," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13. New York, NY, USA: ACM, 2013, pp. 1019–1024. [Online]. Available: http://doi.acm.org/10.1145/2480362.2480557

[16] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? A large-scale study of Stack Overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016. [Online]. Available: https://doi.org/10.1007/s11390-016-1672-0

[17] R. Pokharel, P. D. Haghighi, P. P. Jayaraman, and D. Georgakopoulos, "Analysing emerging topics across multiple social media platforms," in *Proceedings of the Australasian Computer Science Week Multiconference*, ser. ACSW 2019. New York, NY, USA: ACM, 2019, pp. 16:1–16:9. [Online]. Available: http://doi.acm.org/10.1145/3290688.3290720

[18] "Replication Package." [Online]. Available: https://doi.org/10.5281/zenodo.5044270

[19] E. Aghajani, C. Nagy, O. L. Vega-Márquez, M. Linares-Vásquez, L. Moreno, G. Bavota, and M. Lanza, "Software documentation issues unveiled," in *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, J. M. Atlee, T. Bultan, and J. Whittle, Eds. IEEE / ACM, 2019, pp. 1199–1210. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00122

[20] S. Patil and K. Lee, "Detecting experts on quora: by their activity, quality of answers, linguistic characteristics and temporal behaviors," *Social network analysis and mining*, vol. 6, no. 1, p. 5, 2016.

[21] T.-H. Chen, S. W. Thomas, and A. E. Hassan, "A survey on the use of topic models when mining software repositories," *Empirical Softw. Engg.*, vol. 21, no. 5, pp. 1843–1919, oct 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9402-8

[22] "Makar." [Online]. Available: https://github.com/maethub/makar

[23] J. S. Enderle, A. Balagopalan, X. Li, and D. Newman, "Topic modeling tool," 2017. [Online]. Available: https://doi.org/10.5281/zenodo.496150

[24] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002. [Online]. Available: http://mallet.cs.umass.edu

[25] H. M. Wallach, D. M. Mimno, and A. McCallum, "Rethinking lda: Why priors matter," in *Advances in neural information processing systems*, 2009, pp. 1973–1981.

[26] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development issues using Stack Overflow," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 93–96. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487108

[27] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in *Proceedings of the eighth ACM international conference on Web search and data mining*, 2015, pp. 399–408.

[28] M. Hoffman, F. Bach, and D. Blei, "Online learning for latent dirichlet allocation," *advances in neural information processing systems*, vol. 23, pp. 856–864, 2010.

[29] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-Graber, and D. M. Blei, "Reading tea leaves: How humans interpret topic models," in *Advances in neural information processing systems*, 2009, pp. 288–296.

[30] C. Rosen and E. Shihab, "What are mobile developers asking about? A large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9379-3

[31] L. R. Biggers, C. Bocovich, R. Capshaw, B. P. Eddy, L. H. Etzkorn, and N. A. Kraft, "Configuring latent dirichlet allocation based feature location," *Empirical Software Engineering*, vol. 19, no. 3, pp. 465–500, 2014.

[32] S. Beyer and M. Pinzger, "A manual categorization of Android app development issues on Stack Overflow," in *Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution*, ser. ICSME '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 531–535. [Online]. Available: http://dx.doi.org/10.1109/ICSME.2014.88

[33] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen, "Communication in open source software development mailing lists," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 277–286.

[34] E. Miller, "Universal Methods of Design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions," 2012.

[35] "Stack Overflow discussion 379346," accessed August 3, 2020, online. [Online]. Available: https://www.stackoverflow.com/questions/379346

[36] P. Rani, S. Panichella, M. Leuenberger, A. Di Sorbo, and O. Nierstrasz, "How to identify class comment types? A multi-language approach for class comment classification," *Journal of Systems and Software*, vol. 181, p. 111047, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221001448

[37] S. Beyer, C. Macho, M. Di Penta, and M. Pinzger, "What kind of questions do developers ask on stack overflow? a comparison of automated approaches to classify posts into question categories," *Empirical Software Engineering*, pp. 1–44, 2019.

[38] M. Allamanis and C. Sutton, "Why, when, and what: Analyzing Stack Overflow questions by topic, type, and code," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 53–56. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487098

[39] G. Uddin and F. Khomh, "Automatic mining of opinions expressed about apis in stack overflow," *IEEE Transactions on Software Engineering*, 2019.

[40] F. Wen, C. Nagy, G. Bavota, and M. Lanza, "A large-scale empirical study on code-comment inconsistencies," in *Proceedings of the 27th International Conference on Program Comprehension*. IEEE Press, 2019, pp. 53–64.

[41] P. S. Kochhar, "Mining testing questions on Stack Overflow," in *Proceedings of the 5th International Workshop on Software Mining*, ser. SoftwareMining 2016. New York, NY, USA: ACM, 2016, pp. 32–38. [Online]. Available: http://doi.acm.org/10.1145/2975961.2975966

[42] K. F. Tómasdóttir, M. Aniche, and A. van Deursen, "Why and how JavaScript developers use linters," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 578–589.

[43] "Should code comments be formal English or informal English," accessed August 3, 2020, online. [Online]. Available: https://www.quora.com/Should-code-comments-be-formal-English-or-informal-English?

[44] D. Binkley, M. Davis, D. Lawrie, J. I. Maletic, C. Morrell, and B. Sharif, "The impact of identifier style on effort and comprehension," *Empirical Software Engineering*, vol. 18, no. 2, pp. 219–276, 2013.

[45] M. Smit, B. Gergel, H. J. Hoover, and E. Stroulia, "Maintainability and source code conventions: An analysis of open source projects," *University of Alberta, Department of Computing Science, Tech. Rep. TR11*, vol. 6, 2011.

[46] G. Wang, K. Gill, M. Mohanlal, H. Zheng, and B. Y. Zhao, "Wisdom in the social crowd: an analysis of quora," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 1341–1352.

[47] S. K. Maity, A. Kharb, and A. Mukherjee, "Language use matters: Analysis of the linguistic structure of question texts can characterize answerability in quora," in *Eleventh International AAAI Conference on Web and Social Media*, 2017.

[48] M. Neshati, Z. Fallahnejad, and H. Beigy, "On dynamicity of expert finding in community question answering," *Information Processing & Management*, vol. 53, no. 5, pp. 1026–1042, 2017.

[49] S. Geerthik, K. R. Gandhi, and S. Venkatraman, "Domain expert ranking for finding domain authoritative users on community question answering sites," in *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, 2016, pp. 1–5.

[50] B. Mathew, R. Dutt, S. K. Maity, P. Goyal, and A. Mukherjee, "Deep dive into anonymity: Large scale analysis of quora questions," in *International Conference on Social Informatics*. Springer, 2019, pp. 35–49.

[51] G. Bavota, "Mining unstructured data in software repositories: Current and future trends," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 5. IEEE, 2016, pp. 1–12.