API Topics Issues in Stack Overflow Q&As Posts: An Empirical Study

George Ajam
College of Information Technology
University of Babylon
Babylon, Iraq
george@itnet.uobabylon.edu.iq

Carlos Rodríguez
Departamento de Electrónica e Informática
Universidad Católica "Nuestra Señora de la Asunción"
Asunción, Paraguay
carlos.rodriguez@uc.edu.py

Boualem Benatallah
School of Computer Science and Engineering
UNSW Sydney
Sydney, Australia
b.benatallah@unsw.edu.au

***Abstract*—Application Programming Interfaces (APIs) have become one of the key assets within modern businesses, facilitating the linking and integration of intra- and inter-organizational data and systems in the context of complex and heterogeneous technology ecosystems. APIs allow organizations to monetize data, build profitable partnerships and foster innovation and growth. Understanding APIs and their usage are therefore key to building solutions for enabling successful business operations. This paper aims at understanding API topic issues posted on Stack Overflow (SO), a Community Question Answering (CQA) site for programmers. We conduct an empirical analysis on a sample of 400 randomly-selected Q&As threads to help identify API-related issues and their main topics. A thematic analysis performed on this sample reveals eight main topics related to APIs, among which API usage, debugging, API constraints and API security emerged as the major ones. We also exemplify the types of support provided by SO community in addressing each of the identified topics and discuss possible venues on how to further leverage this knowledge.**

***Index Terms*—API topic issues, Stack Overflow, Empirical Study**

I. Introduction

APIs allow developers to reuse software components in the context of software development. An API can be seen as a contract between a provider and consumer of a given functionality [1]. With the rapid growth of APIs and their ever increasing relevance in the support of business operations and integration[1], APIs have become an essential and critical component for software development[2]. Besides enabling the reuse of existing software components, APIs facilitate the exchange of inter- and intra-organizational data, e.g., made available through specific endpoints in the context of web APIs [2], [3]. Furthermore, APIs are regarded now, more than ever, as a key enabler for new and exciting developments, including Blockchain technologies [4], Internet of Things (IoT) [5] and Cognitive Services [6]–[10].

Different resources help developers learn about APIs. For instance, developers may resort to official reference documentation [11], getting-starting guides[3], developers Q&As[4], mailing lists [12], blog posts [13], among other resources. While these resources provide a valuable help to developers dealing with APIs in their daily tasks, not much is actually known and understood about the various API topic issues addressed in such resources. Some attempts to understand topics discussed within these resources (although not necessarily focused on APIs) include the identification of question types within the Q&As of SO [14], the investigation of API security topics posted on GitHub open-source repositories [15], and topics and trends in developer discussions on SO [16]. The identification of API topic issues is key to support the management, exploration and understanding of APIs, the summarisation of key concepts, the discovery of significant topics, among other activities [17].

In this paper, we propose to empirically analyse knowledge related to API topic issues in SO, a widely used programming CQA that is nowadays considered an indispensable resource for facilitating the usage and understanding of APIs [18]. More specifically, we aim to address the following reseach question: *What are the main API topic issues in SO's Q&A posts?* By answering this question, the ultimate goal of our study is to provide insights into topic issues addressed in SO that are unique to APIs. Such insights can help shed light into understanding the main problem areas faced in the usage of APIs, limitations found in API documentation, issues in the design of interfaces, among other aspects relevant to APIs that can affect software development.

The rest of this paper is organized as follows. Section 2 provides background information on the management and quality of API documentation. Section 3 outlines the data collection and analysis techniques employed in our study.

[1]https://www.mulesoft.com/resources/api/connected-business-strategy
[2]https://www.programmableweb.com/news/programmableweb-api-directory-eclipses-17000-api-economy-continues-surge/research/2017/03/13
[3]https://developers.facebook.com/docs/pages/getting-started/
[4]https://stackoverflow.com

G. Ajam, C. Rodriguez, B. Benatallah. "API Topics Issues in Stack Overflow Q&As Posts: An Empirical Study", The XLVI Latin American Computing Conference - CLEI 2020, Loja, Ecuador

Next, Section 4 summarises the findings of our study. Section 5 discusses the implications of our findings and limitations of our study. We close this paper with Section 6 where we present our concluding remarks.

II. Background and Related Work

APIs have become one of the key assets within modern businesses, facilitating the linking and integration of organizations and technology ecosystems. APIs allow organizations to monetize data, build profitable partnerships and foster innovation and growth.[5] The quality of APIs and effective managament of API documentation are therefore key dimensions that need to be considered for a successful business strategy enabled and fueled by APIs. In this section, we explore the literature and research efforts along these two lines of research.

Effective management of API documentation. The management of API documention can be grouped into three categories: (i) API documentation improvement, (ii) API documentation production, and (iii) API documentation problems handling. In the first category, research works focus on recommendation and examples of projects and code, including approaches such as Exemplar [19], PropER-Doc [20] and APIMiner [21]. Works along this line typically aim at augmenting existing API documentation for improvement. *API documentation production*, instead, typically focuses on identifying and producing useful API documentation by leveraging on existing code repositories. Works on this front include the automatic generation of code fragments [22], identification of reusable classes and methods [23], and mining and summarisation of code examples for API usage. In the context of *API documentation problems handling*, researchers focus on issues and failures in API documentation. Work done in this area includes the identification of API documentation problems categories [24], evolution of API documentation, communication and versioning [25], and undocumented exceptions [26].

Quality of API documentation. The quality of API documentation have been explored in the literature along a number of dimensions. We discuss related work under the dimensions of (i) completeness, (ii) understandability, (iii) maintainability, (iv) consistency and (v) correctness.

To improve *completeness* of API documentation, studies investigated tooling to provide examples of API usage such as code examples from unit tests (e.g., [27]) and Java code examples (e.g., [11]). Research under the *understandability* dimension focuses, e.g., on enrichments based on summaries [28] and insights [29] in the context of Q&As, abstraction of the semantics of API calls [30], and enhancement of code examples through code summarisation [21]. While the efforts in the context of *maintainbility* explore the evolution of API documentation during events such as versioning [25], the involvement of developers during documentation revisions [31], and the enrichment of API documentation with up-to-date code examples to support its maintability [11].

On the *consistency* front, efforts focus on detecting and highlighting API issues and inconsistencies. Work along this line includes highlighting inconsistency problems in documentation along with possible solutions [32], detecting errors and inconsistencies in documentations [33], and documentation style suggestions to avoid consistency issues [34]. Finally, the *correctness* dimension focuses on aspects such as bugs generated due to incorrect documentation [35], and support for correct API description through machine-generated documentation [36].

A number of works [16], [37]–[40] have explored the topics and issues covered in programming CQAs such as SO. Yet, to the best of our knowledge, this is the first work that focuses on and qualitatively analyzes API topic issues in the context of programming CQAs. Thus, our paper complements the works discussed previously by unveiling API topics/sub-topics issues that can help understand the main aspects programmers are concerned with when using APIs.

III. Data Collection and Analysis

This section provides details of the data collection and analysis techniques used for conducting our study. We explain how we select our data and the methods we follow for performing the analysis. To analyse the topic issues in SO Q&As posts, we follow the guidelines presented in [41] to develop a thematic map of API topic issues. Next, we provide details on the data collection and the qualitative analysis process.

A. Data Set

We used SO's posts and relied on tags that help us identify API-related posts. The APIs considered in this paper include not only web APIs, but also software libraries and Software Development Kits (SDKs). Tags in SO are used to help categorise Q&As posts [42]. A tag consists of one or more words that group questions into categories[6]. A question can be tagged with no more than five tags in SO[7]. In this study, we first investigate whether the tag "API"[8] may help categorise all API-related questions. We found that SO does not encourage the usage of the tag "API" by itself; instead, it is recommended that members of the community use tags of the form "NAME-API" (e.g., Facebook-Graph-API). Alternatively, if the tag "API" is used by itself, it needs to be accompanied by an additional tag that can help characterise posts in a more precise manner (e.g., API, Facebook-Graph, etc.). We used Stack Exchange Data Explorer (SEDE)[9], an online tool that allows users to write custom queries to retrieve SO data.

Initially, we inquired about tags related to APIs[10], which consists of 550 tags in total, with 338,339 threads overall as

[5]https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/what-it-really-takes-to-capture-the-value-of-apis

[6]https://stackoverflow.com/tags

[7]https://stackoverflow.com/help/tagging

[8]https://stackoverflow.com/tags/api

[9]https://data.stackexchange.com/stackoverflow/queries

[10]http://data.stackexchange.com/stackoverflow/query/746028/all-questions-tagged-with-a-given-tag-and-possibly-other-tags-too-with-score-at

of 11th April 2018. To determine the sample size [43], we employed a 5% margin of error and a confidence interval of 95%, and by using z-score $z = 1.96$. We obtained a recommended sample size of 338.7 posts. We thus rounded this number up to 400 Q&A threads.

We used a query to retrieve 400 randomly selected threads tagged with '%API%', where the "%" wildcard symbol in this query represent any combination of characters. The query above therefore can include threads tagged with tags of the form "NAME-API", "NAMEAPI", "APINAME", "API-NAME" and "API"[11]. We further manually curated the original list of 400 posts to eliminate those containing unrelated tags. For example, we removed posts that contained "api" as part of a word (e.g., "z**api**er-cli", "mkm**api**tem", etc.). As a result, we filtered out 166 posts and ended up with a total of 234 posts for the analysis.

B. Qualitative analysis

We used thematic mapping analysis [41] as the main qualitative method for analyzing the sampled Q&A threads. The guidelines of the thematic mapping analysis suggest that the process starts with data extraction, data coding, and identification of the emerged themes and checking their synthesis for trustworthiness. We refer to the themes emerging from the thematic analysis as API topics. The resulting Q&As from the data sampling step contained a total of 234 posts, each of which was analysed separately following guidelines from [44]. During the analysis we (i) checked the relevance of posts to verify if they indeed discuss APIs and related issues, (ii) checked the attributes of each post (e.g., tags) to further get insights into the post, and (iii) investigated the main topics related to APIs following the steps of thematic analysis [41]. We coded each post to support the formation of the thematic mapping for the topics using a spreadsheet, which we make available on-line.[12]

IV. Findings

We present our findings from the exploratory study of API topic issues. Table I shows the details of our dataset. By following our heuristic for selecting API-related posts (see Section III-A), we included 234 out of 400 randomly selected posts in April 2018. Based on our heuristic, we identified that approximately 2% of SO posts are related to API issues.

The analysis of the selected 234 posts revealed eight main topics related to APIs: *API Definition, API Usage, API Security, API Constraints, Debugging, API Documentation, API Settings*, and *Curation*. We present these topics in Figure 1, which is also available as an interactive, on-line map[13]. We define each topic in Table II and then elaborate on each of these topics and their subtopics in the following subsections.

[11]http://data.stackexchange.com/stackoverflow/query/840515/get-top-400-questions-with-content-tagged-with-something-randomly

[12]https://tinyurl.com/t29d8px

[13]https://tinyurl.com/s7yuywj

TABLE I
SO API POSTS RATIO AND SAMPLING

SO Posts	API related posts	API posts Ratio	Suggested Sample size	Actual Sampling	Selected Posts
16,039,215	338,339	2%	338	400	234

TABLE II
API TOPIC ISSUES DEFINITIONS

Topic	Description
API Definition	Dealing with API specifications, classes and descriptions of methods. On a higher level, finding an API suitable for usage.
API Usage	This topic handles API usage by supporting implementation of features, checking for the possibility of implementation, understanding and seeking better ways to implement a feature using APIs.
API Security	In this topic, the main concern is to handle authentication and authorization to support features implementation and understand such implementation.
API Constraints	Constraints and limitations imposed by the API owner on its usage.
Debugging	Identifying and fixing defects related to using an API method or request (e.g., in Web APIs).
API Documentation	Redirection to reference documentation or reporting issues.
API Settings	Deals with settings related to APIs, the associated working environment and other requirements for using an API.
Curation	Activities performed to improve the content by creating, adding, updating, and removing content (i.e., textual content, code, tags, URIs, error logs) in API-related questions.

A. API Definition

The definition of an API typically includes specifications for object classes, routines, parameters, data structures, and descriptions of methods, usually listed in an API documentation [1]. Developers may find such definitions challenging to understand, unclear, and with incomplete descriptions [24]. In this context, we refer to *API Definition* to the definition of methods, parameters and design patterns (for client code) for the proper usage of APIs. We have under this topic: *Version management, API/Framework recommendation, setting parameters* and *API design patterns*. We discuss each of these next.

API Version Management. This sub-topic includes dealing with resources, classes and methods across different API versions by an API client. An API client code typically binds to a specific version of an API, usually for a specific time window. This binding fails when the client code does

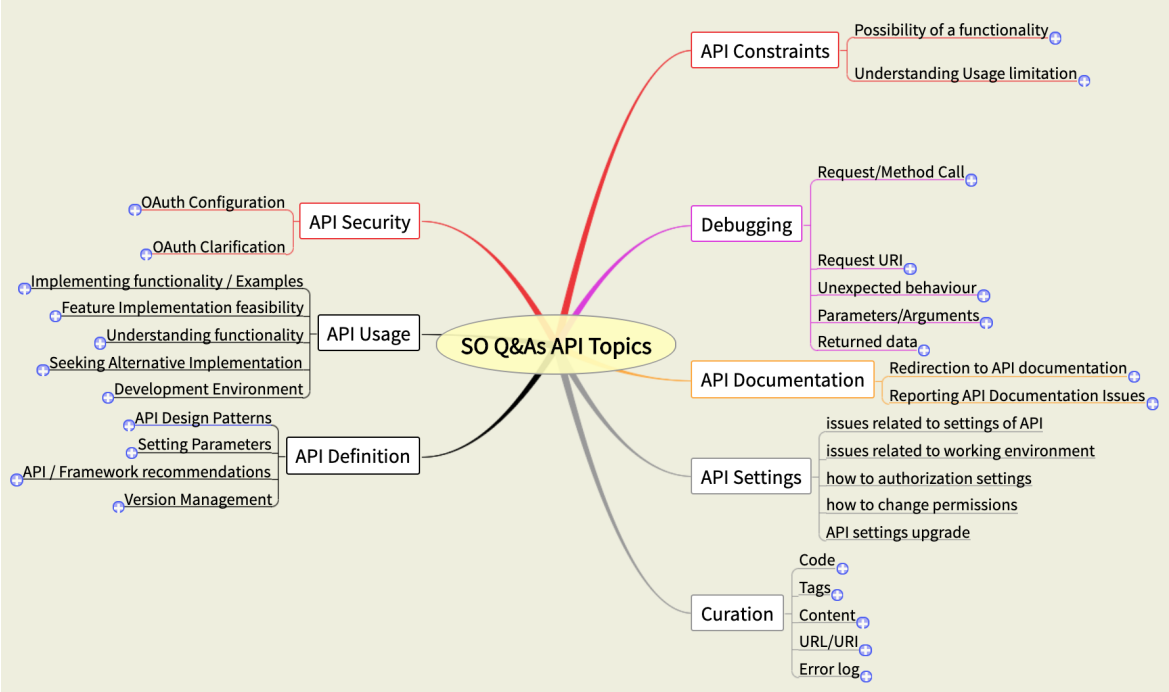

Fig. 1. The main topic issues identified in SO Q&As along with their respective sub-topics. An interactive, online map is available at https://tinyurl.com/s7yuywj

not consider the current version of the API. This sub-topic includes posts with issues related to handling versions of APIs and their appropriateness for the programming language and software libraries at the developer end. For example, when using a specific version of an API, any client code needs to comply with its requirements to get the expected data or functionality. For instance, in question [Q150] (see footnote 12 for a complete list of questions), the reason of an API request issue is found to be related to the usage of an API version that is different from the version on a server (see the answer to question [Q150]). Issues that involve introducing new methods or the deprecation of some methods are considered disruptive to consumers and developers [45].

API/Framework Recommendation. This sub-topic refers to posts related to identifying and recommending suitable framework libraries or APIs, targeting support to specific usage scenarios or programming tasks. For example, a user asked, *"Is there any library (or even better, web service) available which can convert from a latitude/longitude into a time zone?"* [Q335]. The accepted answer, listed on SO, includes some available solutions and suggests a service that the answerer developed to do the exact required functionality. In SO, asking for opinions/recommendations such as for APIs, frameworks and libraries are considered off-topic[14]. This aims at discouraging opinion-based Q&As.

[14]https://meta.stackoverflow.com/questions/255468/opinion-based-questions

Setting up Parameters. While several studies focus on the usage of APIs such as [46]–[49], our analysis shows that developers still find difficulties identifying which API parameter can help them with their intended tasks. SO community can help developers to choose suitable parameters to use. For example, in post [Q242], a user needs to understand parameters for using SQL APIs. An explanation provides details about the functions (e.g., statements parsed, processing, and parameters sent) to fulfil the request.

API Design Patterns. Design patterns are a set of formalised guidelines and practices that developers may follow when designing a software [50]. This sub-topic includes posts to help understand proper design patterns and practices. Issues within this topic are typically related to best practices and decisions in API design. For example, post [Q164] has an explicit request for a design decision about using an object or a collection as a parameter. The community provided suggestions that include best practices as well as advantages and disadvantages for each design decision.

B. API Usage

Posts within this topic report on various usage activities that support the development of software using APIs. API Usage topic includes the following activities: *Features implementation, implementation feasibility, understanding functionality, seeking alternative implementation* and *development environment*.

Features Implementation. This topic represents an implementation of a feature using an API (i.e., part of a software development task using API classes / methods or an API request). In *features implementation*, we find coverage for:

How-to-Use-a-Function: The questioner describes a function and asks how to use it. Most posts include initial code in the question with details about the API used, asking how to implement it. For instance [Q34], a developer needs to implement a function that saves the application state using ASP.NET Web API and an initial code is included. In the answer, a new code example is proposed with specific API methods to help support usage required by the questioner. This subtopic is similar to the question type "How-To-Do-It" identified by Nasehi et al. [14], and represents a significant part of developers' questions. Showing good prior efforts with code may help questions in attracting answers[15].

Handling API Return Data. Handling return data can involve encoding, serialisation and pagination. An example question [Q387], inquiries about how to handle complex JSON response in an API request, and how to populate the values before sending them as an HTTP POST request. The answer suggests the appropriate methods to be used for the intended task.

Getting-Data. Posts within this sub-topic include questions about how to obtain specific values (i.e., how to get the required data from an API). For example [Q240], a questioner asked how to get a specific data field from an API. The answer contains a detailed explanation on how to get the data, which functions can be used, and provides a code example to retrieve the required data field.

Getting-Started and Learning. This subtopic is related to beginners' inquiries and their needs to understand API usage. Posts are typically about API usage, and in most cases it becomes clear from the corresponding answers that these questioners are beginners. Answers contain suggestions on getting-started steps, accompanied by links to API reference documentation or getting-started guides. In [Q347], the questioner needs to understand an example extracted from an API tutorial on how to do web scrapping. The requester further adds parts of the tutorial's code. The corresponding answer provides the class and methods used by the library and adds a quote from the official API documentation, which helps understand key concepts.

Feature Implementation Feasibility. In posts under this subtopic, questioners mainly ask if it is possible to implement a feature using a specific API. Answers typically provide suggestions, which are often accompanied by links to relevant documentation. For example [Q48], a developer asks if publishing drafts is possible using Facebook Graph API. In the response, the answerer shows an alternative solution with a reference to relevant documentation, specifying also which parameters to use in the request. The answerer also complains about the *bad* documentation for the "draft" feature in the official API documentation.

Understanding Functionality. This sub-topic includes posts that are concerned with understanding an implementation, including usage of an API method, the behaviour of a method and clarifications about proper usage of an API. For example, post [Q108] has a request to understand the usage of two web scrapping API requests, and the response to this question is a description of the methods with code example.

Seeking for Alternative Implementation. Here, users ask questions about how to use API methods and the alternative solutions to the task at hand. For example, in post [Q354], the questioner needs to track which window object has keyboard focus, explaining that using *OnIdle()* and *GetFocus()* methods does not seem appropriate. A highly voted answer suggests using *Windows Hooks*, providing documentation link and quote from the documentation along with descriptions of how to do it.

Development Environment. A development environment can be seen as an API sandbox. It allows API users to simulate and test APIs and mimic the requests and responses for testing purposes. This supports the development of applications as if the API is being used in a production environment. For example, in post [Q67], a user of an API faces challenges related to Facebook Ads API sandbox. The corresponding answer includes the steps and directions for using the sandbox testing environment.

C. API Security

Security issues that developers are facing while using APIs are mostly related to authorisation. Authorisation is part of the sign-in process that developers implement in their applications. In this context, OAuth[16] is an example of an open standard (with several implementations in the form of APIs) that is widely utilised for this task. We find two major subtopics within API Security: *OAuth configuration* and *OAuth clarification*.

OAuth Configuration. This sub-topic is concerned with configuring OAuth, more specifically: How to use OAuth, how to get tokens, and how to make proper redirection after the OAuth process is completed, among other questions. For example, in post [Q19], a questioner asks "*How can I redirect to non-secure URL from the successful authentication of Box API? [...]*". The corresponding answer suggests that BOX API OAuth 2.0 needs secure redirection and explains why it is important to send tokens through secure channels.

OAuth Clarification. This sub-topic includes questions and answers that help make OAuth more comprehensible. It differs from the previous sub-topic in that the former focuses more on configuration and implementation aspects, rather than the

[15]https://meta.stackoverflow.com/questions/261592/how-much-research-effort-is-expected-of-stack-overflow-users

[16]https://oauth.net

conceptual understanding of the technology. Here, questioners try to understand concepts related to authorisation using the OAuth protocol and, in some cases, seek for alternative solutions. For example, one questioner asked [Q303] if an API token can be used in multiple applications and the implications thereof. The corresponding answer provides clarifications and discuss the possible side effects (e.g., terms and conditions, throttle limits, etc.).

D. API Constraints

APIs are in many cases (intentionally) designed with certain constraints such as an input/output constraints, pagination and authentication recurrence. This is typically done for performance and other purposes. The *API Constraints* topic includes posts that help understanding such constraints as well as the potential of specifying if a needed functionality is available as a feature by the API or not. We identify two sub-topics here:

Possibility of Functionality. We identified situations in which users seek for an implementation of a feature that is currently not available or fisible with the API of interest. For example, in [Q26], a questioner asked how to change the permission of a user entity, and the answerer replied that such changes are not possible due to missing features by the API. In other posts (e.g., [Q277]), questioners are advised to ask for a feature request addressed to the API owner.

Understanding Usage Limitation. In this sub-topic, we find posts that contain limitations enforced by the API. Users require clarification to understand such limitations. For example, in post [Q391], a user makes HTTP requests and faces error messages. The messages indicate that there is either an insufficient buffer or queue size limitation. The answer to this question suggests that the issue behind this problem are multiple parallel requests, and quotes part of the API documentation with the reference link. The answerer further proposes that the user checks the API terms of service for compliance [Q391].

E. Debugging

This topic covers posts with issues related to debugging. The analysis reveals the following sub-topics: *Debugging request/method call*, *debugging request URI*, *debugging unexpected behaviour*, *debugging parameters/arguments*, and *debugging returned data*. Next, we explain each of these subtopics.

Debugging Request/Method Call. Here, debugging is typically related to an incorrect method invocation or incorrect request. A question with an incorrect method invocation typically includes the description of the issue, provides code snippets, and includes the resulting error message(s). The debugging required consists in resolving the issue for the method to function correctly. For example, [Q17] uses Box-API to upload files but gets an error *"400 Invalid input parameters in the request"*, and the code was posted along with the response. The corresponding answer states that an SDK is available to address this issue, a link to the suggested SDK repository is provided, and a code example is given to show the proper usage of the SDK.

Debugging Request URI. Debugging in this subtopic is related to the errors that occur because of incorrect usage of the API's URI. For example, in [Q148], a user attempts to make a request but the URIs were found to be incorrect. In [Q232], the returned values of an API are empty, and the answerer provides an example on how to properly build the URI so that data gets properly retrieved.

Debugging Unexpected Behaviour. Debugging related to unexpected behaviour is concerned about an outcome of a function that produces wrong output or behaves unexpectedly. For example, in [Q91], Apache Kafka[17], a distributed streaming data API, produced many log files with 512MB in size collectively. The user wonders why such issue happens, and an answer suggests to look at the timestamps retention so that logs get deleted accordingly. Typically, such errors take the form of posts that contain defects with unexpected behaviour, with a solution and explanation that suggest changes to the code.

Debugging Parameters/Arguments. Posts under this subtopic include errors caused by incorrect usage of parameter or method arguments. An error may be related to a field name such as in [Q136]. Similarly, an error may relate to an incorrect parameter as in [Q9]. The answers include explanations with corrected parameters or arguments typically contained in a code example.

Debugging Returned Data. This sub-topic is concerned with issues related to incorrect data returned from an API call or request. For example, in [Q129], a user is not getting the expected return data messages, and the answer suggests a code example supported by a link to a tutorial blog post.

F. API Documentation

API Documentation is one of the many resources that supports API usage. We report on two identified sub-topics: *Redirection to API documentation* and *reporting API documentation issues*.

Redirection to API documentation. This sub-topic includes posts requesting support to implement a basic task, by describing the need, or by asking for help with issues such as errors or implementation difficulties. Answers to these posts include a redirection to an API documentation link. While API documentation link was used in more than one area throughout our analysis, what was clear under this sub-topic is that the answer is full redirection, meaning that the answer is not elaborated with an explanation or example. For example, in [Q254], such redirection is posted as an API Documentation link related to the question. In addition, we have observed

[17]https://kafka.apache.org/

descriptions along with the redirection as well, including quotes of a parts of an API documentation description.

Reporting API Documentation Issues. This sub-topic includes posts with issues related to an appartently incorrect usage of APIs (e.g., incorrect returned data). A further inspection relevead that posts under this category actually relate to errors or typos found in the documentation of the API, similar to what Uddin and Robillard reported in [24]. That is, users faced difficulties using the API because of incorrect API documentation. For instance, in [Q257], a user cites a documentation reference regarding inconsistencies with the method names in the API documentation (e.g., using a specific parameter caused errors). The answerer states that it was a typo and thanks the poster for reporting it.

G. API Settings

APIs require prior settings in some cases before using them, such as enabling specific utilities or subscribing to a specific service plan. In this topic, settings needed to be changed, handled or appropriately enabled before using the API. For example, in [Q7], performing a specific functionality requires having an API developer from the support team to enable it to resolve the issue. In a similar scenario [Q29], settings can be adjusted by the user of the API, and the answer explains how to do it. This topic is different from *Setting up parameters* in that it is related to requirements that enable functionality of the API.

H. Curation

In SO, content is curated in various forms.[18] A crowd of developers are actively creating and curating content, driven by gamification mechanisms [51]. Curating SO Q&As follows the rules set out by the community. We report on curation sub-topics based on the curated elements as follows: *Curating code*, *curating tags*, *curating content*, *curating URLs/URIs* and *curating error logs*.

We find that curation activities related to API programming code examples are *adding code, removing code, formatting code, and general editing for existing code content*. Formatting includes indentation (i.e., adding spaces or tabs) so that the code example is properly structured and presented on SO's website. Another formatting activity involves adding *"back-ticks"* snippets of code so that it is treated as an in-line code when presented with textual content.

Tags curation mainly involves adding or removing API-related tags that belong to a question. Such activity is essential as it categorises API-related questions and makes them more visible. Some developers may watch or search unanswered questions based on specific tags in order to answer these questions. Content curation activities also involve *additions* to the textual content of posts, which include *edits* to the title and body of questions, and *grammar and spelling*.

API-related URL/URIs has also been curated to improve posts. One such curation involves moving from HTTP protocol to HTTPS in links found in posts to help improve security. Finally, we report on *adding/removing error log traces* to support activities that improve understanding issues related to debugging.

[18]https://stackoverflow.blog/2011/02/05/suggested-edits-and-edit-review

V. Discussion

Figure 2 presents the percentage of posts under each API topic. Note that in this figure we do not include *curation*, as curation tasks are pervasive in all sampled posts. Among the topics presented in Figure 2, *API Usage* (40%), *Debugging* (26%), *API Security* (11%) and *API Constraints* (10%) represent the main ones. Other topics are less represented in our dataset. Among these four topics, we furthermore found that the two most covered topics are *API Usage* and *Debugging*. On the one hand, API usage focuses on *features implementation* (i.e., implementing a functionality using APIs), which indicates that SO provides good support for programmers who write programming code using APIs. Indeed, the majority of the support received from SO's community seems to be geared toward *how to use an API*. On the other hand, we found that 72% of the debugging support is devoted to providing help with Web API requests (e.g., REST API calls) or API method calls. Such support comes in the form of help in debugging parameters/arguments, debugging requests on URI and debugging implementions that show unexpected behaviour.

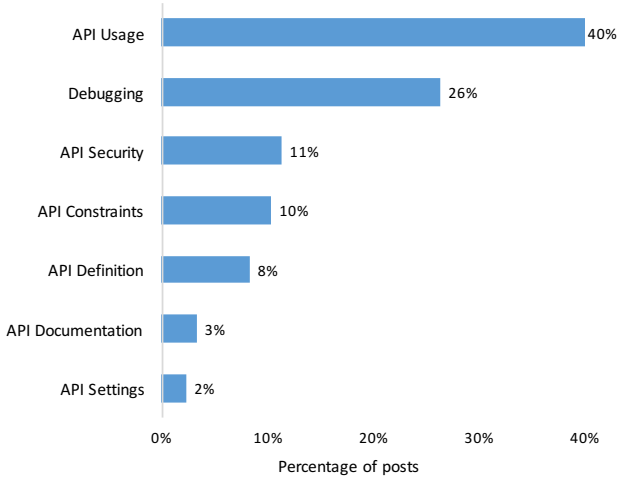

Fig. 2. Percentage of posts per API topics. Note that *curation* is not inlcluded in this chart, as curation tasks are pervasive in all sampled posts.

The findings reported in this study has a number of implications and future directions for both researchers and practitioners. More specifically, the findings can serve as the foundation for approaches and tools that aim at supporting API learning, exploration and discovery, including:

API issues searching and discovery. We identify the need for developing a system that supports the exploration and discovery of API topics and to make related issues searchable. To explore topics and issues related to APIs, we advocate the

development of solutions that allow for topic-based navigation of Q&A posts. Presenting posts categorized under topic issues can facilitate API knowledge exploration and discovery particularly for programmers that are unfamiliar with the target APIs [52].

Collaborative API exploration and testing environments. An interesting direction is also that of providing a shared environment for demonstration and API exploration, as a solution to simulate API sandboxes but in a shared collaborative environment. Such environment can facilitate informing developers about missing features, essential interface contracts and security issues when using APIs. Further empirical investigation along this line can help bring more insights into this area, with the ultimate goal of building and improving tools for learning and using APIs.

Tools to support API features feasibility and constraints awareness. Researchers and practioners in the area of Software Engineering has harnessed CQAs knowledge for a variety of purposes. Yet there are no tools (we are aware of) that supports and facilitates the exploration of (i) the feasilibility of implementing new features within an API (e.g., upon API users formal request), and (ii) constraints imposed on / affecting an API to improve constraint awareness. Such tool can be beneficial for closing the gap between API users and API developers in the context of API constraints and new features.

Notwithstanding the findings reported in this paper, we acknowledge the *limitations* of our study. First, due to the sheer amount of posts in SO (approximately 16 million posts as of April 2018), this study relied on random sampling and heuristics to select API-related posts, which may affect the completeness of the identified topic issues (*recall* perspective). Once posts were sampled, our manual inspection of each post guaranteed that the retained threads indeed contained API-related issues (*precision* perspective). Second, the qualitative nature of this study comes with the risk of introducing human bias during topic issues identification. This limitation was mitigated, as recommended in [41], by seeking agreement of co-researchers, where a Ph.D Student, a Postdoctoral Researcher and a Senior Researcher were involved.

VI. Conclusion

The findings of our study aim at providing a better understanding of current API topics issues in SO. We believe these findings can be useful as a foundation for further studies and tools development that support API documentation, learning and discovery. Future directions for this research includes API topic issues searching and discovery, collaborative and effective API exploration and testing environments, and exploration of API topic issues in other contexts such as software repositories and free and open source software (FOSS) communities.

Acknowledgement. This research was done in the context of the first author's Ph.D. thesis [53] as part of the supervision of his co-authors.

References

[1] W. Maalej and M. P. Robillard, "Patterns of knowledge in API reference documentation," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1264–1282, 2013.

[2] C. Rodríguez, M. Baez, F. Daniel, F. Casati, J. C. Trabucco, L. Canali, and G. Percannella, "REST APIs: a large-scale analysis of compliance with principles and best practices," in *International conference on web engineering*. Springer, 2016, pp. 21–39.

[3] U. Dekel and J. D. Herbsleb, "Reading the documentation of invoked API functions in program comprehension," in *2009 IEEE 17th International Conference on Program Comprehension*. IEEE, 2009, pp. 168–177.

[4] J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, C. Cabanillas, F. Daniel, S. Debois, C. D. Ciccio, M. Dumas, S. Dustdar *et al.*, "Blockchains for business process management-challenges and opportunities," *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, no. 1, pp. 1–16, 2018.

[5] C. Janiesch, A. Koschmider, M. Mecella, B. Weber, A. Burattin, C. Di Ciccio, A. Gal, U. Kannengiesser, F. Mannhardt, J. Mendling *et al.*, "The Internet-of-Things meets business process management: Mutual benefits and challenges," *arXiv preprint arXiv:1709.03628*, 2017.

[6] K. Shaikh, "Eagle-eye view of azure cognitive services," in *Developing Bots with QnA Maker Service*. Springer, 2019, pp. 1–29.

[7] R. Hull and H. R. M. Nezhad, "Rethinking BPM in a cognitive world: Transforming how we learn and perform business processes," in *International Conference on Business Process Management*. Springer, 2016, pp. 3–19.

[8] S. Zamanirad, B. Benatallah, M. C. Barukh, F. Casati, and C. Rodriguez, "Programming bots by synthesizing natural language expressions into API invocations," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 832–837.

[9] S. Zamanirad, B. Benatallah, C. Rodriguez, M. Yaghoubzadehfard, S. Bouguelia, and H. Brabra, "State machine based human-bot conversation model and services," in *International Conference on Advanced Information Systems Engineering*. Springer, 2020, pp. 199–214.

[10] C. Rodriguez, S. Zamanirad, R. Nouri, K. Darabal, B. Benatallah, and M. Al-Banna, "Security vulnerability information service with natural language query support," in *International Conference on Advanced Information Systems Engineering*. Springer, 2019, pp. 497–512.

[11] S. Subramanian, L. Inozemtseva, and R. Holmes, "Live API documentation," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 643–652.

[12] X. Chen and J. Grundy, "Improving automated documentation to code traceability by combining retrieval techniques," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE, 2011, pp. 223–232.

[13] C. Parnin and C. Treude, "Measuring API documentation on the web," in *Proceedings of the 2nd international workshop on Web 2.0 for software engineering*, 2011, pp. 25–30.

[14] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming Q&A in StackOverflow," in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2012, pp. 25–34.

[15] M. Zahedi, M. Ali Babar, and C. Treude, "An empirical study of security issues posted in open source projects," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[16] 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.

[17] A. Hindle, N. A. Ernst, M. W. Godfrey, and J. Mylopoulos, "Automated topic naming to support cross-project analysis of software maintenance activities," in *Proceedings of the 8th Working Conference on Mining Software Repositories*, 2011, pp. 163–172.

[18] F. M. Delfim, K. V. Paixão, D. Cassou, and M. de Almeida Maia, "Redocumenting APIs with crowd knowledge: A coverage analysis based on question types," *Journal of the Brazilian Computer Society*, vol. 22, no. 1, p. 9, 2016.

[19] C. McMillan, M. Grechanik, D. Poshyvanyk, C. Fu, and Q. Xie, "Exemplar: A source code search engine for finding highly relevant applications," *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1069–1087, 2011.

[20] L. W. Mar, Y.-C. Wu, and H. C. Jiau, "Recommending proper API code examples for documentation purpose," in *2011 18th Asia-Pacific Software Engineering Conference*. IEEE, 2011, pp. 331–338.

[21] J. E. Montandon, H. Borges, D. Felix, and M. T. Valente, "Documenting APIs with examples: Lessons learned with the apiminer platform," in *2013 20th working conference on reverse engineering (WCRE)*. IEEE, 2013, pp. 401–408.

[22] T. Gvero and V. Kuncak, "Interactive synthesis using free-form queries," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 689–692.

[23] S. Thummalapenta and T. Xie, "Spotweb: Detecting framework hotspots and coldspots via mining open source code on the web," in *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2008, pp. 327–336.

[24] G. Uddin and M. P. Robillard, "How API documentation fails," *IEEE Software*, vol. 32, no. 4, pp. 68–75, 2015.

[25] S. Sohan, C. Anslow, and F. Maurer, "A case study of web API evolution," in *2015 IEEE World Congress on Services*. IEEE, 2015, pp. 245–252.

[26] M. Kechagia and D. Spinellis, "Undocumented and unchecked: Exceptions that spell trouble," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 312–315.

[27] S. M. Nasehi and F. Maurer, "Unit tests as API usage examples," in *2010 IEEE International Conference on Software Maintenance*. IEEE, 2010, pp. 1–10.

[28] L. Guerrouj, D. Bourque, and P. C. Rigby, "Leveraging informal documentation to summarize classes and methods in context," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 639–642.

[29] C. Treude and M. P. Robillard, "Augmenting API documentation with insights from stack overflow," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 392–403.

[30] Y.-C. Wu, L. W. Mar, and H. C. Jiau, "Codocent: Support API usage with code example and API documentation," in *2010 Fifth International Conference on Software Engineering Advances*. IEEE, 2010, pp. 135–140.

[31] B. Dagenais and M. P. Robillard, "Creating and evolving developer documentation: Understanding the decisions of open source contributors," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, 2010, pp. 127–136.

[32] F. F. Correia, A. Aguiar, H. S. Ferreira, and N. Flores, "Patterns for consistent software documentation," in *Proceedings of the 16th Conference on Pattern Languages of Programs*, 2009, pp. 1–7.

[33] H. Zhong and Z. Su, "Detecting API documentation errors," in *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications*, 2013, pp. 803–816.

[34] L. Thominet, "Building foundations for the crowd: Minimalist author support guides for crowdsourced documentation wikis," in *Proceedings of the 33rd Annual International Conference on the Design of Communication*, 2015, pp. 1–10.

[35] M. F. Zibran, F. Z. Eishita, and C. K. Roy, "Useful, but usable? Factors affecting the usability of APIs," in *2011 18th Working Conference on Reverse Engineering*. IEEE, 2011, pp. 151–155.

[36] P. J. Danielsen and A. Jeffrey, "Validation and interactivity of web API documentation," in *2013 IEEE 20th International Conference on Web Services*. IEEE, 2013, pp. 523–530.

[37] M. Allamanis and C. Sutton, "Why, when, and what: analyzing Stack Overflow questions by topic, type, and code," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 53–56.

[38] P. K. Venkatesh, S. Wang, F. Zhang, Y. Zou, and A. E. Hassan, "What do client developers concern when using web APIs? An empirical study on developer forums and Stack Overflow," in *2016 IEEE International Conference on Web Services (ICWS)*. IEEE, 2016, pp. 131–138.

[39] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, "Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow," *Georgia Institute of Technology, Tech. Rep*, vol. 11, 2012.

[40] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, "Classifying stack overflow posts on API issues," in *2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 2018, pp. 244–254.

[41] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *2011 international symposium on empirical software engineering and measurement*. IEEE, 2011, pp. 275–284.

[42] W. Mo, J. Zhu, Z. Qian, and B. Shen, "SOLinker: Constructing semantic links between tags and URLs on StackOverflow," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2016, pp. 582–591.

[43] G. D. Israel, "Determining sample size," University of Florida, Florida Cooperative Extension Service, Fact Sheet PEOD-6, 1992.

[44] K. Charmaz, *Constructing grounded theory: A practical guide through qualitative analysis*. sage, 2006.

[45] S. Wang, I. Keivanloo, and Y. Zou, "How do developers react to restful API evolution?" in *International Conference on Service-Oriented Computing*. Springer, 2014, pp. 245–259.

[46] M. Ichinco, W. Y. Hnin, and C. L. Kelleher, "Suggesting API usage to novice programmers with the example guru," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017, pp. 1105–1117.

[47] W. Wang, H. Malik, and M. W. Godfrey, "Recommending posts concerning API issues in developer Q&A sites," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 224–234.

[48] N. Wu, D. Hou, and Q. Liu, "Linking usage tutorials into API client code," in *Proceedings of the 3rd International Workshop on Crowd-Sourcing in Software Engineering*, 2016, pp. 22–28.

[49] K. Zimmerman and C. R. Rupakheti, "An automated framework for recommending program elements to novices (n)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 283–288.

[50] D. Riehle and H. Züllighoven, "Understanding and using patterns in software development," *Tapos*, vol. 2, no. 1, pp. 3–13, 1996.

[51] H. Cavusoglu, Z. Li, and K.-W. Huang, "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*, 2015, pp. 171–174.

[52] C. E. Grant, C. P. George, V. Kanjilal, S. Nirkhiwale, J. N. Wilson, and D. Z. Wang, "A topic-based search, visualization, and exploration system," in *The Twenty-Eighth International Flairs Conference*. Citeseer, 2015.

[53] G. Ajam, "Quality of Application Programming Interfaces Documentation and Topics Issues Exploration." Ph.D. dissertation, University of New South Wales, Sydney, Australia, 2019.