IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 6, 2019, accepted April 28, 2019, date of publication May 2, 2019, date of current version May 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2914429

Toward Empirically Investigating Non-Functional Requirements of iOS Developers on Stack Overflow

ARSHAD AHMAD[1,2], CHONG FENG[1], KAN LI[1], SYED MOHAMMAD ASIM[3], AND TINGTING SUN[1]

[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[2]Department of Computer Science, University of Swabi, Ambar 25000, Pakistan
[3]Department of Statistics, University of Peshawar, Peshawar 75190, Pakistan

Corresponding author: Chong Feng (fengchong@bit.edu.cn)

This work was supported in part by the National Science Foundation of China (NSFC) under Project U1636203, and in part by the National Key Research and Development Program of China (NKRDPC) under Project 2017YFB1002101.

ABSTRACT Context: Mobile application developers are getting more concerned due to the importance of quality requirements or non-functional requirements (NFR) in software quality. Developers around the globe are actively asking a question(s) and sharing solutions to the problems related to software development on Stack Overflow (SO). The knowledge shared by developers on SO contains useful information related to software development such as feature requests (functional/non-functional), code snippets, reporting bugs or sentiments. Extracting the NFRs shared by iOS developers on programming Q&A website SO has become a challenge and a less researched area. Objective: To identify and understand the real problems, needs, trends, and the critical NFRs or quality requirements discussed on Stack Overflow related to iOS mobile application development. Method: We extracted and used only the iOS posts data of SO. We applied the well-known statistical topical model Latent Dirichlet Allocation (LDA) to identify the main topics in iOS posts on SO. Then, we labeled the extracted topics with quality requirements or NFRs by using the wordlists to assess the trend, evolution, hot and unresolved NFRs in all iOS discussions. Results: Our findings revealed that the highly frequent topics the iOS developers discussed are related to usability, reliability, and functionality followed by efficiency. Interestingly, the most problematic areas unresolved are also usability, reliability, and functionality though followed by portability. Besides, the evolution trend of each of the six different quality requirements or NFRs over time is depicted through comprehensive visualization. Conclusion: Our first empirical investigation on approximately 1.5 million iOS posts and comments of SO gives insight on comprehending the NFRs in iOS application development through the lens of real-world practitioners.

INDEX TERMS Non-functional requirements (NFRs), quality requirements, iOS, Latent Dirichlet allocation (LDA), Stack Overflow.

I. INTRODUCTION

Requirements Engineering (RE) plays a vital role in the success of any software development process. RE is quite challenging, and there are many activities associated with it that are required to be adequately addressed in every software development life cycle. Requirements need to be adequately elicited, stated, verified & validated, and maintained as needed [1]–[4]. In recent years, the RE community started considering the user feedback available on different social media and online platforms as one of the potential sources of user requirements for fostering open innovation [5]–[9]. Chesbrough [10] defined open innovation as "*a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as they look to advance their technology*." Open innovation paradigm has been recognized as a novel strategy for software development organizations to benefit from the sharing of novel ideas and embracing value creation processes across and outside the software development organizations

The associate editor coordinating the review of this manuscript and approving it for publication was Zhanyu Ma.

boundaries. Some of the available sources of open innovation include social media technologies, online platforms such as the Stack Overflow Q&A community [11], Twitter, issue tracking systems, and mobile application stores like Google's Play Store, and Apple's Play Store [12]. Stack Overflow Q&A community enables software developers to participate in social learning and sharing innovative ideas [13].

Typically, software requirements are of two types: functional requirements (FRs) and non-functional requirements (NFRs) or quality requirements. Concerning the first type, new FRs can be elicited directly from a user through software feature requests [14]. The second type of software requirements (NFRs or quality requirements) can be extracted from the user content shared on different social media platforms like Stack Overflow Q&A site, Twitter, and feedback on different mobile application stores may be of interest since users are directly influenced by different NFRs/quality characteristics, e.g., usability, performance efficiency, and security. Stack Overflow is the most popular and widely used online Q&A forums by software developers around the globe for learning and sharing knowledge on a diverse range of software development related topics. These Q&A are asking help on some problem, reporting a problem/bug in a development tool and requesting a new feature (functional or non-functional requirement) missing in the existing development tools or applications which is ignored in previous research [15]. It is highly probable that the content shared on Stack Overflow posts contain statements about mobile application development, tools, and product qualities [12], and will ultimately help software organizations in fostering open innovation.

The research effort on NFRs or quality requirements is significant, as they are vital to the success of software product development. These NFRs are the architectural drivers [16], and inadequately addressing them will mostly result in project failure and increase in rework cost [1], [4]. Thus, NFRs should be addressed in the early stages of any software development to avoid any underlying problems. However, eliciting an entirely complete and precise set of NFRs or quality requirements is challenging [17], [18].

The recent years have witnessed enormous growth in usage of mobile devices. Consequently, this rapid interest has drawn mobile application developers' attention too recently. The research shows that mobile application development is entirely different from traditional software development due to diversity in development practices, tools, evolving user needs, and platforms [19]–[21]. Some of the research studies considered the issues faced by mobile application developers (e.g., [19], [22– [24]), and others have focused on general software developers issues (e.g. [11], [25]–[28]). However, all of these studies are either too broad or lacks explicitly classification of the iOS mobile application development issues with the ISO9126 quality model.

The discussions made on Stack Overflow represent the real views and needs of software developers across the globe. The posts data extracted from Stack Overflow encompass millions of various posts shared by numerous software developers and professionals. There are millions of posts and comments available on Stack Overflow. However, we have used the data from 31st July 2008 till 31st August 2017 in this paper. More specifically, we merely focused on iOS mobile application development and extracted data tagged with ''iOS'' approximately 525K posts and 925K comments in total. There are numerous challenges associated with analyzing such large-scale textual content repository. For instance, manual analysis is not possible, and the majority of the conventional data mining techniques are not so capable. Thus, the well-known statistical topic model Latent Dirichlet Allocation (LDA) [29], can assist in comprehending the unstructured natural language text data. We applied topic modeling LDA as a means to summarize them.

Previously, topic modeling method has been widely applied to summarize large corpora in certain domains including Software Engineering [19], [22], [25], [30]. Some of the research works on topic analysis also depicts that topics extraction can be helpful to comprehend software maintenance activities [31], [32] better. The topics mined from the repository summarize the main concepts in the repository, for instance, source code, text descriptions, and commit comments. The statistical topic model LDA can identify the topics of the repository. It generates multinomial distributions of the words to characterize the topics.

As the topics extracted are considerably abstract and challenging to comprehend fully, we require suggesting suitable labels for these topics to use it efficiently in discussion and analysis. The extracted topics made sense to real-world practitioners and matched their view of what has occurred [33]. We annotate them with the list of NFRs since the topic trends mostly corresponded to NFRs [34]. Through that, we determine the focus and needs of the iOS mobile application developers and represent visualizations of NFRs that iOS mobile application developers discussed during the development activities. Managers can also understand the topic-related activities of the iOS application developers more conveniently.

In this paper, we mine the topics of the repository using the statistical topic model LDA, and afterward annotate the topics with different NFRs labels using an available wordlist. Then, we examine the NFRs in all iOS discussions and specific iOS development technologies or categories. The NFRs analysis in all iOS discussions comprised of hot NFRs, unresolved issues, and evolution trend. The NFRs analysis in certain iOS technologies or categories comprises of hot NFRs, hard NFRs, and the trend of NFRs difficulty. We split the repository into time-windows when mining the topics of the repository. We divide the repository into 30 day periods (one month) and afterward applied the statistical topic model LDA to each of these time-windows. We partitioned the repository into time-windows rather than considering it as a whole since many topics are pretty local. Besides, this enabled us to examine the evolving trend of the NFRs and difficulty over time. For annotating the NFRs, we use the quality standard of

ISO9126 [35], which defines six high-level NFRs including; maintainability, functionality, portability, efficiency, usability, and reliability.

For our analysis, we utilize the iOS posts data from Stack Overflow. Besides, we also differentiated between iOS posts were having comments (a combination of information sources) or without comments (single information source) to match the outcomes between the two different settings. In this empirical research study, we set out to determine whether user content shared on Stack Overflow can also be a useful source of statements to support the elicitation of NFRs or quality requirements about iOS mobile application development. We specifically restrict the scope of our study to iOS mobile application development only. We have formulated the following research questions which will be addressed in this empirical study:

RQ1: What are the most important non-functional requirements discussed in all iOS discussions and specific iOS technologies or categories on Stack Overflow?

RQ2: Which non-functional requirements questions remain unanswered the most in iOS Stack Overflow posts?

RQ3: What is the trend of the non-functional requirements over time in all iOS Stack Overflow posts?

RQ4: What are the most difficult non-functional requirements the software developers face in specific iOS technologies or categories in Stack Overflow posts?

RQ5: How does the non-functional requirements difficulty in different iOS specific technologies or categories changes over time?

The research questions aim to identify the critical NFRs or quality requirements discussed on Stack Overflow related to iOS mobile application development. We use a massive scale of iOS posts data available on Stack Overflow to investigate not only the essential NFRs along with their trend but also the common problems faced by iOS developers. Since thousands of experienced software developers daily use Stack Overflow, their discussions trends mostly represent the current needs of users and market trends. This will ultimately help 1) iOS developers to know what are the most critical NFRs and issues that need to be addressed first so that they can plan for them accordingly, 2) the evolution of NFRs and developers interests will aid iOS platform providers in providing more desired development support (e.g., offer a new API), 3) the evolution of NFRs trend will also help iOS developers, managers and vendors in comprehending the usage history of their products, 4) fostering open innovation in software development organizations, and 5) assist software engineering academics and industry in identifying the problematic areas for iOS developers that need further research and attention.

The remainder of this paper is organized as follows: Section 2 describes the related work, Section 3 describes in detail the planning and execution of the experimental approach used for this study. The analysis of the results of the study is explained in Section 4. Section 5 discusses the validity threats of this study. Finally, Section 6 provides the conclusion and point out avenues for future research.

The current empirical research work is the extension of our previous research work [36]. The main extensions in this study are the four refined investigations to examine the quality requirements or NFRs in specific iOS technologies or categories, which enable us to know iOS developers interests at a deeper level:

- We assessed the hot quality requirements or NFRs in specific iOS technologies or categories, the associations, and dissimilarities between them.

- We examine the correlation between quality requirements or NFRs significance and time in specific iOS technologies or categories.

- We explored the approach to characterize the difficulty of quality requirements or NFRs and assessed the challenging domains of quality requirements or NFRs the programmers come across in different specific iOS technologies or categories.

- We assessed the evolving trends of quality requirements or NFRs difficulty with time in specific iOS technologies or categories.

II. RELATED WORK

In this section, we shortly summarize the most relevant related work to this study in four different categories: the studies generally focusing on different aspects on Q&A website Stack Overflow; app store analysis focusing on user feedback analysis; the various studies of NFRs; and the diverse use of statistical topic model LDA aimed to study the emphasis and trends in software engineering data.

A. STACK OVERFLOW

There are number research efforts that have investigated different aspects of Stack Overflow. Ahmad et al. [15] recently presented the first comprehensive literature survey on mining Stack Overflow. They briefly summarized the different work on Stack Overflow about software development into two categories ''SO Design and Usage'' and ''SO applications.'' Treude et al. [11] categorized SO questions through qualitative coding and thoroughly examined the pivotal role of SO in software development. Yang et al. [37] performed a study by explicitly exploring the questions related to security on SO. They used LDA topic model tuned using Genetic Algorithm, aimed to group diverse security-related Q's based on their texts. The results revealed that questions related to security on SO cover a diverse range of topics, i.e., web security, mobile security, cryptography, software security, and system security were the most discussed topics in the questions posted on SO. Pinto and Kamei [38] investigated SO posts for categorizing posts related to some refactoring tools. Fontão et al. [39] assessed 1,568,377 technical questions from SO those were related to Android, iOS, and Windows Phone platforms to identify the activity level of developers and hot topics discussed. In contrast, our work aims to identify the issues and most essential NFRs discussed only in iOS posts. We then evaluated the identified NFRs with the ISO9126 quality model which is missing in their work.

Martinez and Lecomte [40] investigated what mobile developers primarily discuss when they develop and maintain cross-platform mobile applications. Specifically, they remained focus on Xamarin framework by using two datasets of Q&A sites: Xamarin Forum and Stack Overflow. Mamykina et al. [41] examined the design features of Stack-Overflow. Asaduzzaman et al. [42] investigated the reasons behind questions are left unanswered on SO. Some of the significant reasons revealed are: unable to find suitable experts, unclear questions, and examples without code snippets. Bosu et al. [43] and Bazelli et al. [44] examined the different personality traits of SO users and mechanism for building a reputation on SO community. Bajaj et al. [28] investigated the misunderstandings and common issues among web developers through using SO posts. Jin et al. [45] studied specific gamified metrics linked with the response time of posts on Stack Overflow.

The work of Goderie et al. [46] presented a tag-based approach to forecasting the response time in SO posts. Calefato et al. [47] investigated and proposed how SO users can enhance the quality of their contribution to have more chances of acceptance. Novielli et al. [48] examined the idea of a sentiment analysis tool to identify the sentimental expressions in the posts of SO. The work by Anderson et al. [49] examined the dynamics of the long-lasting value of questions on Stack Overflow. The works of Anderson et al. [50], Grant and Betts [51], and Marder [52] examined significant aspects of user performance after getting badges, and specifically for evaluating how badges can push users to change their behavior on SO. Slag et al. [53] investigated one-day flies users to know why these users contribute only once on SO. Nasehi et al. [54] manually conducted a qualitative assessment to explore the salient features of perfect code examples in SO posts.

Honsel et al. [55] evaluated common myths about SO posts to discover whether the perception of software developers is accurate. Chowdhury and Hindle [56] proposed an approach to classify off-topic SO posts in programming-related internet relay chat channels. Li et al. [57] empirically investigated to identify the diverse needs and problems software developers face. Parnin et al. [58] empirically studied how SO enables crowd documentation for three popular APIs, namely, Android, GWT, and Java. Linares-Vásquez et al. [22] examined SO posts to classify the hot issues related to mobile application development. However, our work focuses on the perspective of non-functional requirements present in SO posts, to get detailed insights to better comprehend the identified non-functional requirements through mobile developers' eyes. Besides, our work also differs from them as we focus on more diverse technology categories to determine more aspects of iOS mobile development activities.

B. APP STORE ANALYSIS

There are several works done on extracting and analyzing user feedback on different app stores (i.e., Google Play Store, Apple iOS). One of the comprehensive literature survey done by Martin et al. [59] which systematically classifies the work done on different app stores for different software engineering purposes. Harman et al. [60] assessed different technical and business aspects of mobile apps by mining app features from the app descriptions discussed on app stores. Iacob and Harrison [61] retrieved and grouped app feature requests or requirements from the user reviews on the app store using some linguistic rules and statistical topic model LDA. The work of Galvis Carreño and Winbladh [62] also extracted and summarized the different user review made on the app store through using LDA topic model.

Pagano and Maalej [63] examined the useful feedback of users extracted from app reviews productive for app developers. Other works done are on feature-based sentiment analysis of app reviews [14], [64], [65], identifying user satisfaction in the app stores through similar words or phrases with a predefined dictionary [66], identifying incorrectly rated app reviews [67], categorizing the types of complaints by users and assess how complaints affect the ratings on the Apple iOS App Store [68], and the empirical investigation of association between the success of mobile apps, and the change and fault-proneness of the APIs [69]. All of these works extracted and assessed user feedback available on app stores for different purposes, i.e., identifying features requests, bugs, or classifying user feedback to improve the quality of future release of the apps. However, our work is focused on identifying NFRs or quality requirements present in iOS posts on Stack Overflow.

C. NON-FUNCTIONAL REQUIREMENTS

The work of Chung and Nixon [70] considered to be the first work that examined how to handle non-functional requirements in the software development process. Eventually, they conclude that a non-functional requirements framework proposed in [71] would be quite useful for software developers and users. Chung et al. [72], by reviewing on dealing with non-functional requirements conducted a well systematic and noticeable exploration on non-functional requirements in the Software Engineering domain. The work of Mairiza et al. [73] focused on the non-functional requirements on three vital aspects, comprising; definition and terminology, kinds, and non-functional requirements in several types of systems and diverse application domains. The work by Doerr et al. [18] conducted an industrial study on non-functional requirements via a hierarchical quality model. The emphasis of Glinz [74] remained on presenting a comprehensive survey on the definitions of non-functional requirements and identified their issues, and eventually came up with some notions for overcoming these issues/challenges.

The work by Umar and Khan [75] assessed numerous methods, practices, and frameworks of non-functional requirements suitable for software development. The research efforts of Galster and Bucherer [76] emphasized on identifying and classifying non-functional requirements for specific service-oriented systems. Ameller [77] investigated how different software architects consider non-functional

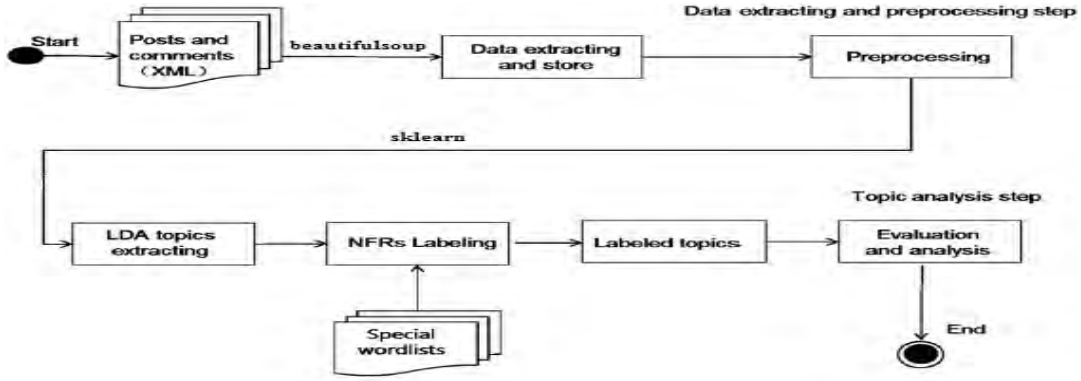


FIGURE 1. Overview of the steps followed.

requirements in practice. Later on, Ameller et al. [78] concerning model-driven development (MDD) combined non-functional requirements into the MDD process. The work of Kugele et al. [79] presented a unified model-driven development process for the deployment of certain real-time systems. The work of Ahmad et al. [80] remained on presenting an approach for modeling and verification of both functional and non-functional requirements. The study of Borg et al. [81] revealed that most software development organizations emphasis on functional requirements while non-functional requirements are hard to elicit effectively. The work by Damm et al. [82] developed a rich component model that combines functional and non-functional requirements based on the UML.

The research work by Cleland-Huang et al. [83] presented a unique approach to identify and categorize the non-functional requirements by requirements documents. The work by Eckhardt et al. [84] discussed the possibility of integrating non-functional requirements in seamless modeling. In another study by Eckhardt et al. [85], concerning the difference between the functional requirements and non-functional requirements, revealed that most of the ''non-functional'' requirements are not non-functional. Besides, they also found that many supposed non-functional requirements can be tackled similarly like functional requirements. The work of [26] focused on identifying non-functional requirements in different development technologies. Noei et al. [86] work investigated the association of both device qualities and app qualities with the user-perceived quality of different Android apps available on Google Play Store by examining 20 device features, for instance, CPU and the display size, and 13 app features, for instance, code size and UI complexity. However, our work focus of research is different from all these efforts. We conduct an empirical study on non-functional requirements through mining programming Q&A website Stack Overflow using topic analysis. We thoroughly analyzed all iOS posts on SO for assessing non-functional requirements focus, level of difficulty and their evolving trends with the time, which is left unexplored in prior research.

D. LATENT DIRICHLET ALLOCATION (LDA)

LDA is amongst the favorite techniques used in the domain of Software Engineering successfully [87]–[89]. Barua et al. [25] proposed a technique to discover and assess the diverse topics from the textual content of SO by using LDA. The results revealed useful insights about topics related to different technologies, for instance, mobile application development, web development, version control systems to C# syntax and data management. Afterward, the different works of Linares-Vásquez et al. [22] and Rosen and Shihab [19] precisely focused on mobile application development related discussions made on SO using LDA. Similarly, the work of Allamanis and Sutton [90] explored topic modeling analysis to relate the questions with different software development concepts and identifiers. The common thing in our work and theirs is using the LDA model, however, is dissimilar from them due to focus and the kind of information analyzed. We remained focused on identifying and analyzing non-functional requirements iOS developers are concerned about based on SO posts and discussions.

III. PLANNING AND EXECUTION

In this section, we describe how we carried out our study in three steps. At first, we extracted the iOS posts data from Stack Overflow, and then applied some preprocessing steps on the extracted data. Then, we applied the topic model Latent Dirichlet Allocation [29] to extract the topics of the corpus. In the last step, we match and label the topics with the identified NFRs through the wordlists defined by Hindle et al. [34], especially suitable for the domain of software engineering. All the steps and the overall process followed in our empirical study is depicted in Figure 1.

A. STEP 1: EXTRACTING AND SELECTING SO POSTS

To address the research questions of our study, we extracted the posts and comments from 31st July 2008 up to 31st August 2017 provided by a programming/social Q&A website Stack Overflow[1]. The Q&A on Stack Overflow

[1] https://archive.org/details/stackexchange

TABLE 1. Questions categorization.

Category	Count	Related Stack Overflow tags
IPhone	100620	Iphone, iphone-sdk-3.0
Objective-C	159358	Objective-c, objective-c-literals
Swift	116401	Swift, swift4
Xcode	77900	Xcode, xcode-tools
Cocoa	24513	Cocoa, reactive-cocoa-3
Cordova	9678	Cordova, cordova-plugin-file
Core-data	14661	Core-data, core-data-migration
Ipad	18624	Ipad, iPad-mini
Uitableview	40485	Uitableview,uitableviewrowaction
Uiview	22520	Uiview, uiviewcontroller

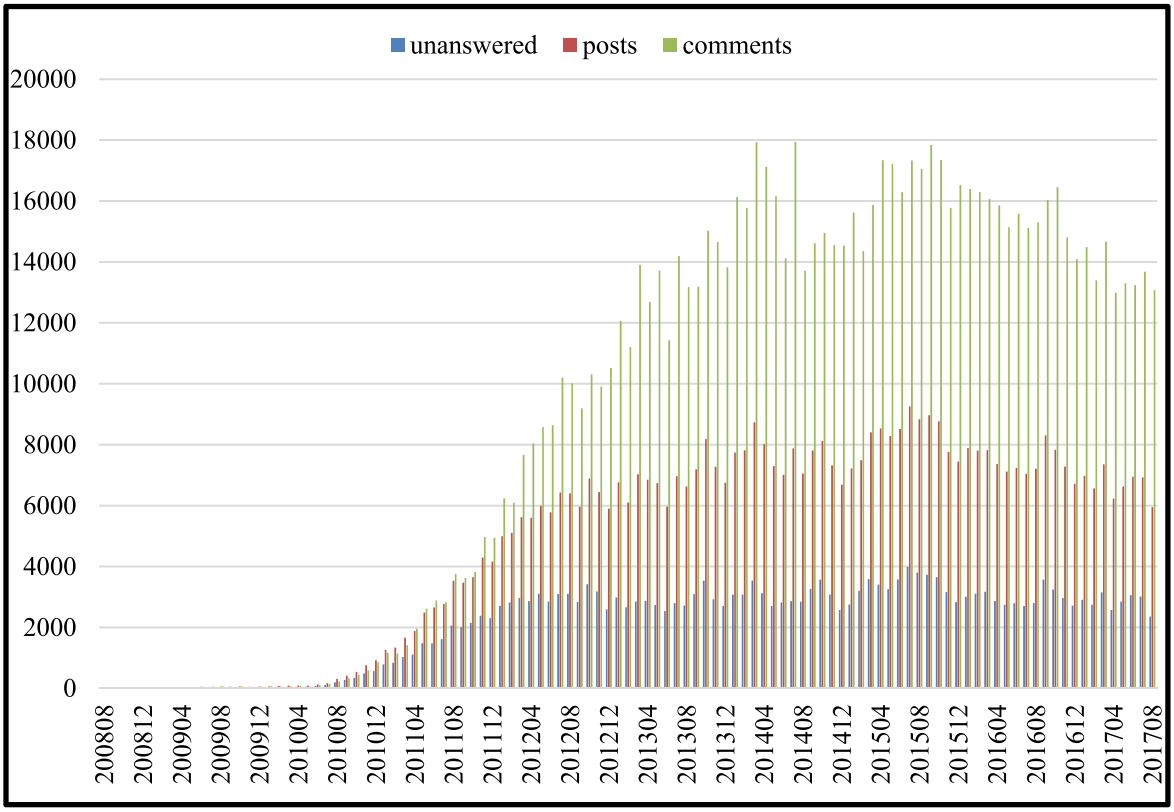


FIGURE 2. Overall dataset used of each month.

consists of a diverse range of questions about software development including mobile application development. These Q&A discussed by developers can be seeking a solution to a problem, knowledge sharing and reporting missing feature in some development tool. The original posts data provided by Stack Overflow is in XML format and have much redundant information. Thus, the authors used the Python library Beautiful Soup[2] and developed an automatic method to filter and extract/select only those posts tagged "iOS," approximately about 525K posts and 985K comments in total. For instance, a sample (Post ID #10848) Stack overflow posts storage in the "Posts.xml" is illustrated in Figure 4. As it is already known that the questions posted on the Stack Overflow website must be tagged according to the desired category, e.g., iOS, Android, and Java. To address the RQ1, we mainly used two types of the corpus: the first is the "title" & "body" of iOS posts combined with the "text" of the comments and the other only has the "title" and "body" of the iOS posts. We compare the outcomes between them, i.e., "title" & "body" of iOS posts combined with the "text" of the comments and "title" and "body" of the iOS posts without the text of the comments.

The second type of corpus is the category data extracted from posts according to Table 1 following [90]. For addressing RQ2, we only extract the "title" and "body" of the unanswered questions from iOS posts totaling approximately

[2]https://www.crummy.com/software/BeautifulSoup/

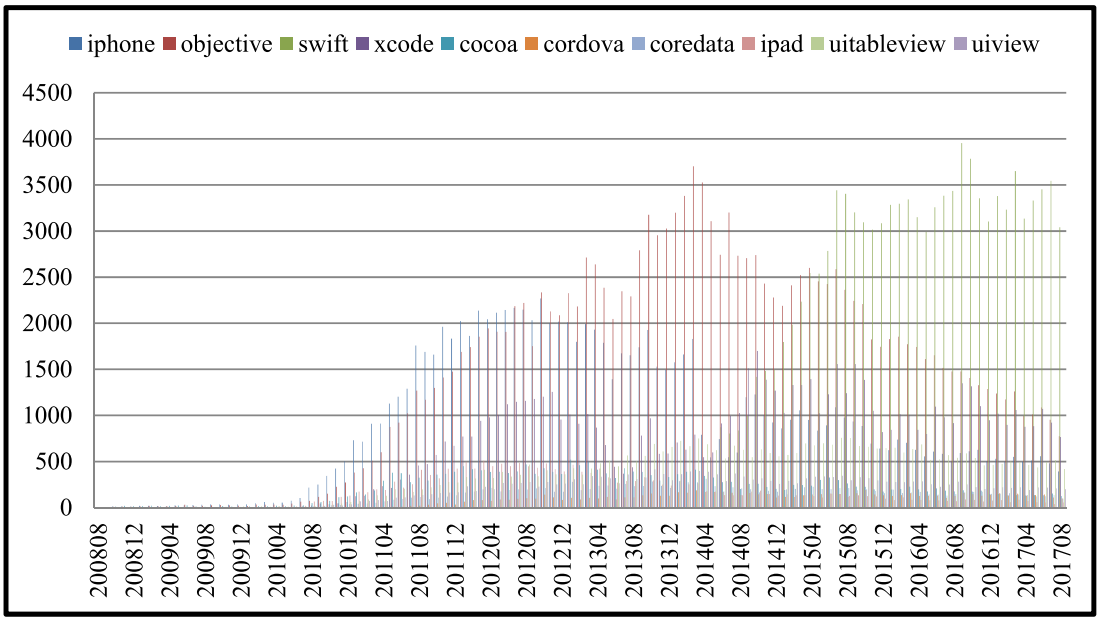


FIGURE 3. Category dataset of each month.

228K. We assume the question as an unanswered question if there is no answer found for the said question. For addressing RQ3, we utilize both the "title" and "body" of iOS posts and the "title" and "body" of the unanswered questions with the aim to examine the trends of quality requirements or NFRs in all iOS discussions, and utilize the posts category data to examine the trends of quality requirements or NFRs in specific iOS technologies or categories. Besides, to address RQ4 and RQ5, the dataset we utilized is also the category data mined from iOS posts. Figure 2 and Figure 3 depict the overall data set and the category dataset of different technology used each month respectively. They depict the details of the data used of each month (period), the x-axis represents the months, and the y-axis represents the number of posts or comments, the highest among them reaches approximately 18K.

After the desired posts data are successfully extracted, we preprocess the data following these two steps. First of all, we remove all those periods (months) which have posts less than 50, since fewer posts are unusable for the sake of analysis. For example, in August 2008, there are only three posts. Afterward, to further refine the information in the posts data we performed tokenization (e.g., removed punctuations), stop words removal (e.g., removed "a," "the" and "is"), and case unification respectively using Python language. All these steps were applied to remove unnecessary information from the posts data.

B. STEP 2: LDA TOPIC MODELING

In this research study, we use and construct the topic model LDA by Python sklearn [91]. The topic model LDA is applied to extract the topics of our corpus. LDA has been successfully applied and one of the most suitable techniques for identifying the topics of discussions in various natural language text documents. Besides, it has also been successfully applied for various purposes in Software Engineering domain, for instance, software change message classification [92], software defect prediction [93], bug localization [88], and component assignment [87]. In LDA topic model, the topic represents the conditional probability distribution of words in a particular vocabulary. LDA uses not only the word frequencies but also the co-occurrences of frequencies in the text documents to build a model of semantically related words [94]. LDA generates semantically related topics when it identifies that there are sets of words that are likely to co-occur frequently in the text documents of the corpus. It is quite often that the words found in a topic are mostly semantically related. For instance, the topics (mouse, click, drag, right, left) and (user, account, password, authentication) represent a group of words that frequently co-occur in the text documents of a particular corpus [94]. Given a set of certain documents of a corpus, the LDA identify the topics and topic assignment for each document present in the corpus through machine learning algorithms.

Applying LDA, it needs specific inputs, i.e., the desired number of topics parameter K, the desired number of iteration N to be carried out, and the Dirichlet parameters α and β. The number of topics K is a user-defined parameter which regulates the granularity of the identified topics. α is a hyper-parameter that regulates how much fuzziness is permitted in a topic's distribution across the words. β is the hyper-parameter which regulates how much fuzziness is allowed in a topic's distribution across the documents of a corpus. For our experimental work, we selected the number of topics parameter $K = 20$ for each of the specified periods since the same words from different topics are not so frequent when the value of $K = 20$. However, the value of $K = 20$ is not necessarily the best choice but proved to be an appropriate value for NFRs analysis as reported in [31], [34]. Besides,

we did not change the default settings for N = 1000, α = 0.05, and β = 0.05. In our experiment, the outcome of the LDA is a matrix M where rows represent the K topics of posts or comments, and the columns represent the words of the topics respectively.

C. STEP 3: LABELING OF TOPICS WITH NFRs

To do NFRs analysis, we annotate the extracted topics with NFRs labels by using the ISO9126 quality model as the taxonomy of quality requirements or NFRs. We lack evidence to claim that ISO9126 quality model is the only correct and comprehensive standard available. Nevertheless, the ISO9126 quality model is the most commonly practiced software quality model at present. Thus, we deem it enough representatives to use for this research. We linked each of the quality requirement or NFR with a list of keywords, known as wordlists. The wordlist used in this study is the exp2; especially suitable for the domain of software engineering [34] is shown in Table 2. The wordlists are not projects specific, i.e., independent which are derived from Boehm's quality model [95] and McCall's quality model [96] respectively. We match the words of the extracted topics with the words in the wordlists. If a match is identified between them, then the topic is labeled with the corresponding NFRs or quality requirement. In case no match is identified between them, then the topic is labeled with ''none'' since the topic is not related to any of the quality requirement or NFRs. Nevertheless, the extracted topics can also be labeled with one or more quality requirement or NFR*s*. To better comprehend the labeling process Table 3 illustrates some exemplary topics, NFRs labeling, and the sample posts which possess these topics as the most frequently occurred topics.

D. STEP 4: NFR's METRIC

From the previous step, we acquired the labeled topics to analyze the NFRs. For addressing the research questions, we then define the NFRs metric called R_{NFRs} to specify the rate of different NFRs. This metric calculates the proportion of topics labeled by the respective NFRs and is given as:

$$\mathrm{R}_{NFRs} = \frac{1}{K \times M} \sum_{x=1}^{M} N_x \tag{1}$$

K represents the exact number of topics extracted by our LDA model in each period (per month). M represents the total number of months that exists in our corpus. N_x represents the number of topics labeled with the respective NFRs in the xth (particular) month. For instance, let's assume that the total number of topics in each month is 20, the total number of months is 5, and the number(s) of topics labeled with an NFR (e.g., reliability) from the 1st month to 5th month are 11, 9, 5, 7, and 8, the reliability has an R_{NFRs} metric of 40%. This means that 40% of all the extracted topics contain the reliability NFR. The R_{NFRs} metric enables us to calculate the relative popularity of a particular non-functional requirement.

E. STEP 5: VALIDATING THE CORPUS

For assessing the automated annotated results, four Ph.D. students in software engineering were invited to do the task of labeling the topics manually as a validation set. The annotation was accomplished in two steps: first of all the four annotators annotated the data from January 2016 till to March 2016 to assess the inter-rater reliability among them. Secondly, each of the four annotators labeled the data (January 2016-December 2016) for three months without annotating each other's annotations as a validation set. The participants looked at the extracted topics of each period (month) and the words of each topic. Then, the participants suggested the suitable label (using one or more NFRs from ISO9126) to the topic based on their knowledge and expertise in the Software Engineering domain. Nevertheless, the participants can also label the extracted topics with ''none'' if they deem there is no NFRs related or linked to the topics. Besides, all of the participants did not annotate each other's annotations. During the labeling task, the participants also utilize the original data as supplementary information associated with the extracted topics being annotated. Moreover, we are quite confident that the manual labeling of topics performed by the participants is correct since they all have enough background and expertise in the Software Engineering domain.

IV. ANALYSIS OF THE RESULTS

A. THE ACCURACY OF THE EVALUATION

To assess the accuracy of our NFRs labeling, we primarily use the well-known metrics of recall and precision rates for our study. We chose one-year post data from January 2016 to December 2016 as the testing set and ran our approach on it. Then, we compare the outcome with the results generated through the manual validation set. The calculation criteria for recall and precision rate are given in Equation 2 and 3 respectively.

$$\text{Recall Rate} = \text{Ndetected/Ntotal} \tag{2}$$

$$\text{Precision Rate} = \text{Ndetected/Ndetectedall} \tag{3}$$

where $N_{detected}$ represents the number of precise NFRs labels (i.e., the NFRs or quality requirements label corresponds to the manual annotation), N_{total} represents the whole number of the manual NFRs labels in our testing set, $N_{detectedall}$ represents the whole number of NFRs labels (both correct and incorrect) generated in the experimental results through our automatic approach. For instance, if our approach labels a topic with usability, reliability, and portability, and in the manual validation set the participants labels the topic as usability, reliability, and functionality. Then, in such case the value of $N_{detected}$ is 2 (usability and reliability), the value of N_{total} is 3 (usability, reliability, and functionality), and finally, the value of $N_{detectedall}$ is 3 (usability, reliability, and portability). After calculating, the values of recall rate are 2/3 approximately 66.7%, and the precision rate is 2/3 approximately 66.7% respectively.

TABLE 2. NFR's and their related wordlists.

Label	Associated terms
Maintainability	maintainability, modular, decentralized, encapsulation, dependency, interdependent, understandability, modifiability, modularity, maintainable, maintain, stability, analyzability, changeability, testability
Functionality	functionality, accuracy, correctness, vulnerability, secure, accurate, vulnerability, vulnerable, trustworthy, malicious, policy, buffer, overflow, secured, certificate, exploit, compliant, functionality, practicality, functional, suitability, interoperability, accuracy, compliance, security
Portability	portability, transferability, interoperability, documentation, internationalization, i18n, localization, l10n, standardized, migration, specification, portability, movability, movableness, portable, installability, replaceability, adaptability, conformance
Efficiency	efficiency, optimization, fast, slow, faster, slower, penalty, factor, sluggish, optimize, profiled, performance, efficiency, efficient, "time behavior," "resource behavior."
Usability	usability, flexibility, interface, screen, user, friendly, convention, human, default, click, guidelines, dialog, ugly, icons, ui, focus, feature, standard, convention, configure, menu, accessibility, gui, usability, serviceability, serviceableness, usableness, useableness, utility, usefulness, serviceable, usable, useable, learnability, understandability, operability
Reliability	reliability, failure, error, redundancy, fails, bug, crash, stable, stability, integrity, resilience, dependability, dependableness, reliability, reliableness, responsibility, responsibleness, dependable, reliable, maturity, recoverability, "fault tolerance."

<row answercount="18" body=" Does anybody know if it's possible, and how, to programmatically send a
SMS from the <code>iPhone</code>, with the official SDK / Cocoa Touch? "commentcount="3"creationdate="2008-08-14T09:51:06.207"favoritecount="177"id="10848" lastactivitydate="2017-06-28T14:47:07.993"lasteditdate="2015-12-17T12:08:55.033" lasteditoruserid="1280373" ownerdisplayname="Marco" posttypeid="1"
score="463" tags="<ios><objective-c><cocoa-touch><sms>"
title="How to programmatically send SMS on the iPhone?"
viewcount="261050"></row>

FIGURE 4. An example of iOS post.

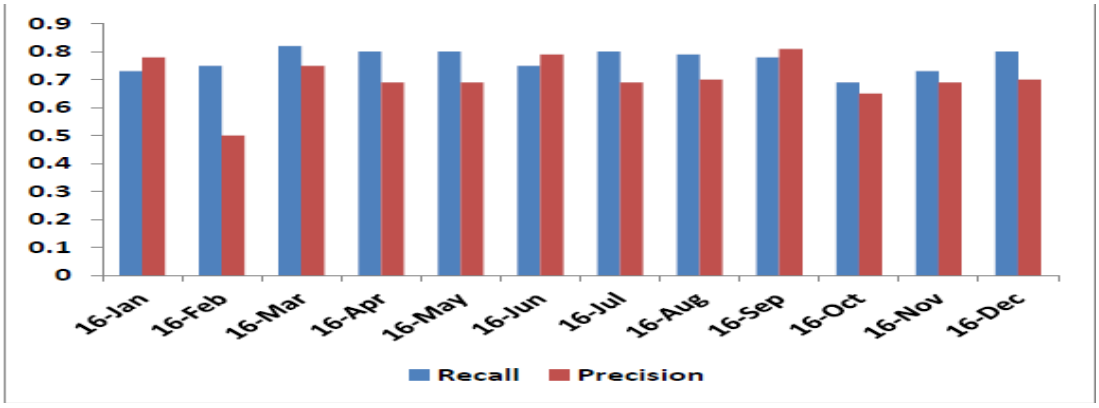


FIGURE 5. Calculated percentage of recall and precision rate of NFRs labeling.

Figure 5 depicts the calculated recall rate and the precision rate for each period (month) of our results. It is evident in Figure 5 that the highest recall rate is 82%, and the precision rate is 81% respectively of our study results averaging approximately 77% and 70.33% respectively.

We performed an inter-rater reliability test to minimize the annotators' NFRs labeling bias. We employed non-parametric Kendall's coefficient of concordance (W) [97] to assess the inter-rater agreement between four annotators using the three months labeled data by four annotators.

TABLE 3. Sample identified topics, NFRs labeling, and posts.

Topics (Top 50 words)	NFR's Labeling	Sample posts
user facebook app ios login api access profile account sdk email post password share identifier users token link twitter provisioning session logged log application error https authentication developer dialog username friends sign fb message graph code keychain friend permission check integration activity info sharing deprecated valid indicator bundle add unable……	Usability, Reliability	**#20182446:** localstorage reliability with phonegap.i'm making a phonegap app that needs to store some user data. on the initial app startup, the user will be asked to type in a url. because the url may potentially be long, i wish to save it on the user's device so the he doesn't need to re-enter the entire string every time he starts up the app.initially, i was planning on using localstorage for this. however, i've heard that localstorage doesn't save data very permanently. it would greatly hurt my app's usability if the user had to type in the url more than once every month or so.should i use sqlite instead of localstorage for this purpose, or is localstorage reliable enough on most mobile devices for this kind of usage?
xcode error project build ios version app target beta issue pod failed code sdk command install https running errors installed module cocoapods swift release maps compile posts pods debug fast projects application building solve add unity fine archive slow linker ipa targets apple built exit clean option macos update unable……	Efficiency	**#45341915:** efficient way to display multiple arrays of data into single tableview section.i have a data model as below:the model will be fetched and parsed from api, i use another class data as a sub-model to form model. and i need to display the model object into a single table view section in the order: field1 -> field2 -> [field3] -> [field4]what i am planning to do is as below:}as my point, it's a lot inefficient, since every time cellforrowat is called, it has to calculate the array size again and again, but i have to check the array to display them in the pre-defined order above. however, that is the only way currently in my mind.i would like to improve the code for better efficiency, may be the way the data should be stored in the model(instead of an array or something like that...) or how to populate data properly into the table view. please help to point me out where should the code be improved and how.thanks **#44921600:** will there be efficiency issue if access to keychain frequently?i have a tableview that row height needs to adjust dynamically according to the value stored in keychain, so comes the problem.just wondering how to verify it quantitatly.

Kendall's coefficient of concordance (W) value has a scale between 0 and 1, where 0 depicts full disagreement, and 1 depicts full agreement. In our study, Kendall's coefficient of concordance (W) for the labeling of the topics was 0.71, which depicts a substantial level of agreement [98] between the four annotators. Based on the depicted substantial agreement between all of the annotators, our labeling of topics is reasonably consistent. Nonetheless, based on the experiences gained from annotators of this study, we suggest other researchers have proper training and discuss the whole annotation process comprehensively to get more significant agreement among all of the annotators' annotation.

B. RESULTS OF RQ1

For addressing the first research question, we primarily investigated which NFRs are discussed the most and less frequent to discover the hot NFRs the mobile application developers emphasis on. We study this from two main aspects. Firstly, we determine the hot-NFRs in all iOS discussions. Then, we investigate the hot-NFRs in specific iOS technologies or categories.

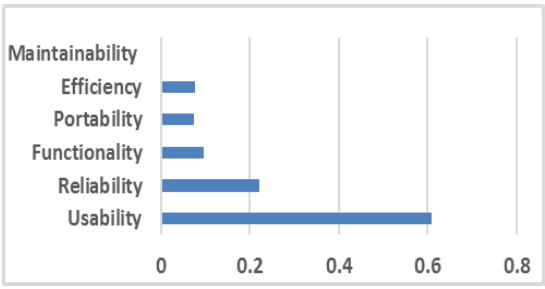


FIGURE 6. Rate of distribution of different NFRs in posts only.

Figure 6 depicts the rate of six different quality requirements or NFRs using the iOS posts data. The x-axis represents

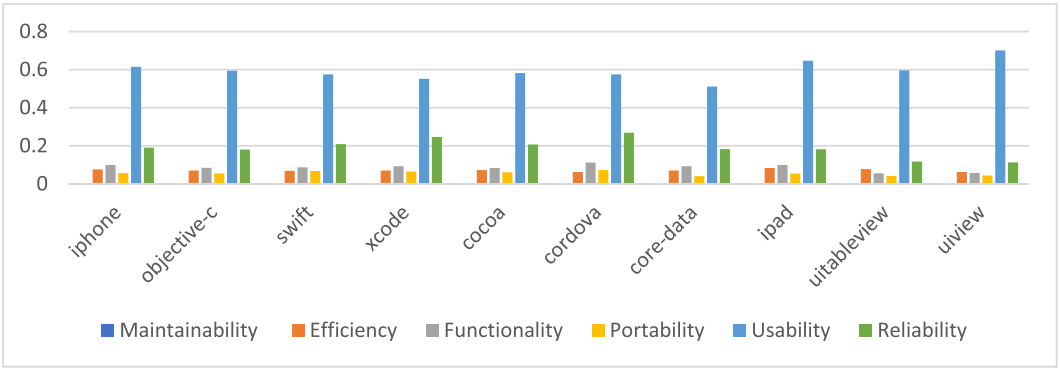


FIGURE 7. Rate of distribution of different NFRs in specific technologies.

the rate or value of the R_{NFRs}. The y-axis represents the six respective quality requirements or NFRs. It is evident in Figure 6 that the labels with the highly frequent topics are usability, reliability, and functionality. Efficiency and portability are less frequent NFRs or quality requirements. We did not see the maintainability NFRs. This trend of six different quality requirements or NFRs shows that mobile application developers are more concerned about usability, reliability, and functionality. It also reveals that they come across several problems of usability, reliability, and functionality when developing iOS applications. On the other hand, they are less concerned or face fewer problems of efficiency, portability, and maintainability during iOS application development. It specifies that the majority of the iOS developers' worldwide lean towards discussing the usability of mobile applications rather than maintainability. It also advises the worldwide mobile applications vendors should comparatively give more significance to the usability of the mobile applications. Besides, we also examine the rate of different NFRs in posts with user comments. The results revealed that the rate of different quality requirements or NFRs is almost similar to the results of using posts only, i.e., the highly frequent topics in descending order are usability, reliability, functionality, efficiency, portability, and maintainability (none). Moreover, it means that the generated results are identical exploring the topics with and without comments.

As most of the mobile application developers' work on one or more iOS specific development technologies, they might be concerned in the certain NFRs that emphasis in the specific iOS development technologies or categories. In addition to identifying the hot NFRs in iOS discussions, we also extracted the data of certain iOS technologies or categories to discover the associations and dissimilarities of the NFRs focusing on various sub-domains. We primarily were interested in recognizing whether some iOS development technologies or categories discuss certain quality requirements comparatively more often than the others. It is useful for project managers when deciding the initial design goals, or to track the development objectives at a particular stage. Figure 7 illustrates the rate of six different NFRs in different iOS technologies or categories. It is evident from Figure 7 that the hot NFRs label is usability, followed by the reliability and functionality NFRs in the ten different iOS technologies or categories. The NFRs labels efficiency and portability are comparatively less discussed topics. These statistics depict that the majority of the iOS mobile application developers focus mostly on the usability aspect, and almost none on the maintainability aspect, irrespective of the development technologies or categories they are involved in. The iOS mobile application developers tend to give more value or emphasis on usability more than any other NFRs while developing mobile applications. The comparative trends of the six NFRs in ten different iOS categories are almost identical, and the gap among the rates of each NFR on various iOS technologies or categories is also minor. It ultimately recommends that software quality issue also is of similar concern amongst the different technologies or categories. Besides, it is also evident that the relative trends of six different NFRs in Figure 6 and Figure 7 are almost identical. These results indicate that the difference in technology does not affect six different NFR to be or not to be hot over time.

C. RESULTS OF RQ2

To address the RQ2, we primarily focused on all unanswered iOS questions on Stack Overflow to investigate the unsolved critical domains. It in return will be more helpful for the iOS application developers to highlight the most challenging issues they face during application development. Figure 8 depicts the distribution rate of six quality requirements or NFRs about all iOS unanswered or unaddressed questions. It is evident in Figure 8 that the most frequent topics remain unresolved or unanswered are labeled with usability, reliability, functionality, and portability. The less frequent topics remain unanswered are labeled with efficiency and maintainability being the least frequent among all. It means that the iOS developers are facing continuous critical problems in handling usability and reliability issues. It warrants that more focus should be put on usability and reliability of iOS development since developers often unable to handle them. The issues of functionality and portability are comparatively less frequent but still needs attention to have successful iOS applications development. The least frequent are efficiency

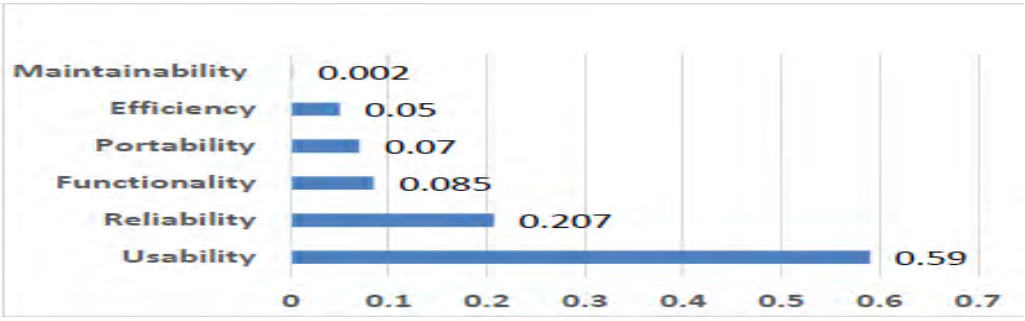


FIGURE 8. Rate of distribution of different NFRs in unanswered posts or questions.

and maintainability problems in iOS development developers face, or they can better handle it easily those issues. In the future, more research is needed in this area to investigate in detail the nature of all those critical issues so that the academia and industry should come up with possible solutions.

D. RESULTS OF RQ3

To address RQ3, we only use posts data because the outcome of RQ1 determined that using posts alone and using posts along with the comments have the same results. Through our approach, we label the extracted topics of iOS posts, and revealed that most of the topics are labeled with one NFR is approximately 62.39%, more than one NFR are approximately 27.93%, and approximately 9.68% are labeled with ''none.'' Table 4 illustrates the percentages of different NFRs labeling.

TABLE 4. Non-functional requirements labeling percentages.

	Percentage
One NFR	62.39%
Multi-NFRs	27.93%
None	9.68%

Figure 9 (a) & (b) depicts the gray-scale image of the six different quality requirements or NFRs frequencies over the timeline. The cell corresponds to 30 days period. The higher intensity or deep color of a grid cell represents the lowest label frequency, i.e., less count of all NFRs over the passage of time. The lighter intensity of grid cell represents the higher frequency of NFRs over time. Figure 9(a) & (b) not only depicts the evolution of each of the six different NFRs visually with time but also depicts the trend of hot or not hot NFRs in a particular timeframe. Figure 9(a) depicts the outcomes of the iOS posts, and it is evident that almost all of the quality requirements or NFRs evolve except maintainability. Nevertheless, the trends of the NFRs are entirely different from one another. The trend of usability is almost entirely stable over the whole period having the highest frequency. The frequency of reliability is higher at the start but then its trend decrease over time. It is also evident that efficiency, functionality, and portability frequency trends are up and down with time. The frequency of maintainability is least among all and stays constant from the start until the end of the period.

Figure 9(b) depicts the outcomes of the unanswered iOS questions on SO. The frequency trend of efficiency, functionality, portability, and reliability is quite low at the start but then increase with time. The frequency trend of usability is the highest at the start and remains quite stable except a slight decrease is observed at the end. The frequency trend of maintainability is again the least (almost none) among all NFRs. To summarize the findings of both Figure 9(a) and Figure 9(b), it is evident that the trend of reliability, portability and functionality are interestingly growing not only in the iOS posts but also in the unanswered iOS questions. The trend of usability is having the highest frequency and is stable on both iOS posts and unanswered iOS questions. All these findings hints that reliability, portability, and functionality will raise the attention of the iOS developers and the usability will most probably stay hot in the coming years.

Besides, we also determine the correlation between the significance of software quality requirements or NFRs and time in specific iOS technologies or categories. We are primarily interested in investigating whether some NFRs are of more interest as the time evolves in specific iOS technologies or categories. For addressing this question, we equally split the whole dataset into eight non-overlapping frames (or windows) by time. Each of the frame (or windows) has 12 months posts data, to identify the NFRs trends from coarser time granularity in specific iOS technologies or techniques. For instance, Frame 0 has the feature requests data generated from August 2008 to July 2009; Frame 1 has the feature requests data generated from August 2009 to July 2010, and the list goes on till Frame 7 has the feature requests data generated from August 2016 to July 2017. Then, the rate of labeled topics is acquired.

Figure 10 shows that the rate of different NFRs over time in specific iOS technologies or categories. From Figure 10, it is evident that the rates of NFRs (especially usability, reliability, and functionality) are generally increasing with time for most of the iOS technologies or categories. Besides, we also explore the relationship between the six different NFRs and time by calculating their Spearman's

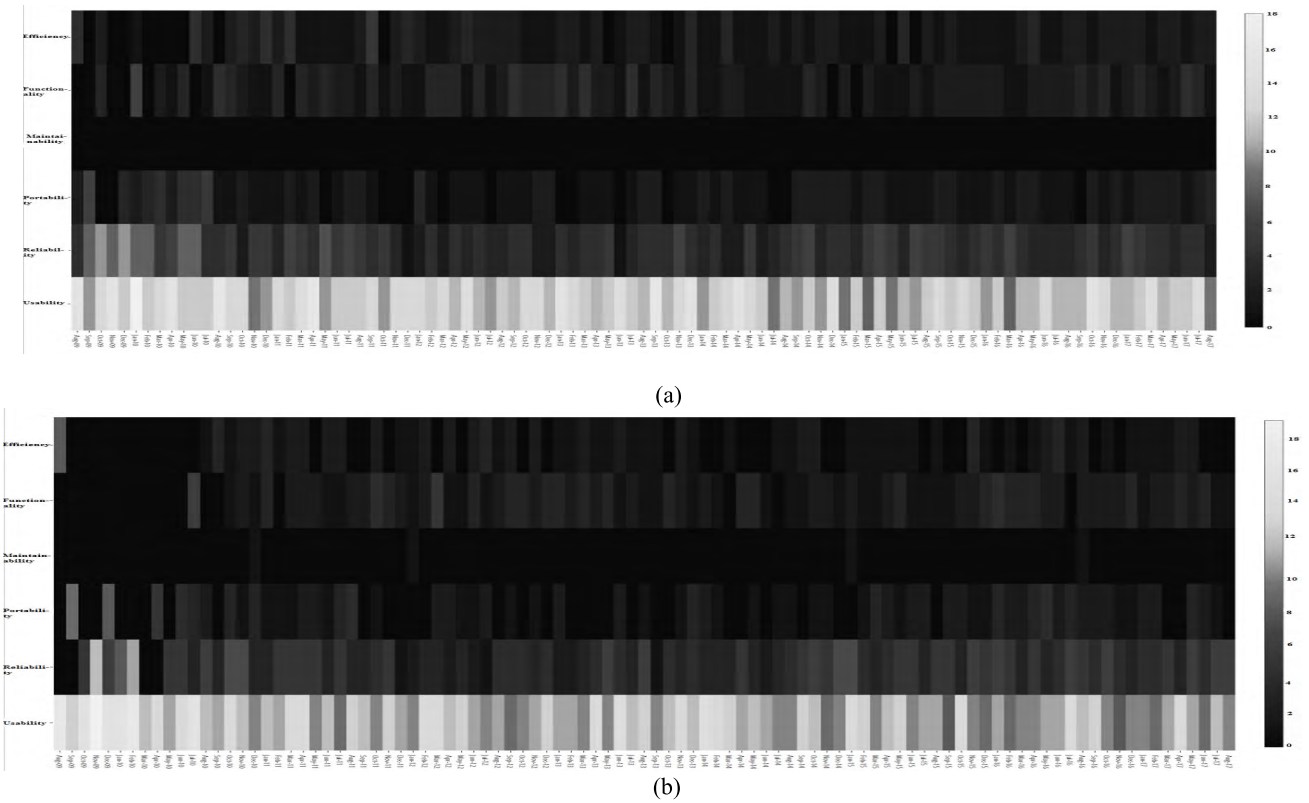


FIGURE 9. Rate of frequencies of NFRs over time. (a) iOS Posts. (b) Unanswered iOS Posts.

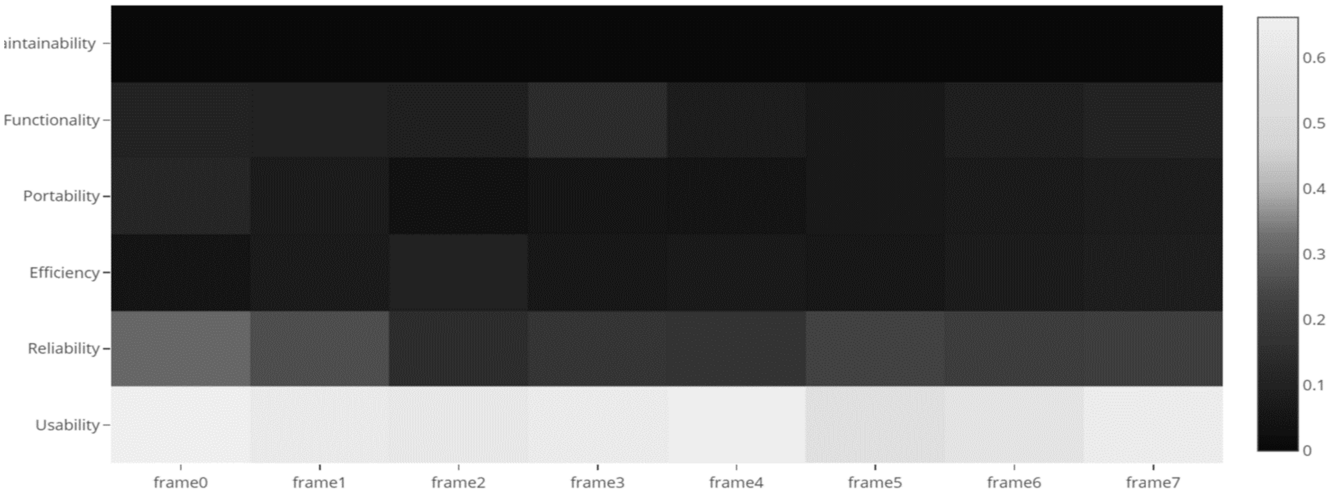
coefficients. The purpose of calculating the Spearman's coefficients is to know the correlation between the importance of six different NFRs and the time. The outcomes of Spearman's coefficients are given in Table 5. The Spearman's rank correlation coefficients denoted by $r_s$and values are constrained to ± 1, i.e., values closed to the ± 1 are deemed as strongly correlated [98]–[100]. In the iOS technologies or categories Cocoa, Cordova, iPad, iPhone, Objective-C, Swift, and XCode the usability NFRs are strongly correlated. In the iOS technologies or categories Core-data, efficiency NFRs is strongly correlated, and in UItableview portability NFRs is strongly correlated. Moreover, in all of the ten iOS technologies categories, the usability, reliability, functionality and efficiency NFRs are comparatively more correlated than the rest of the portability NFRs. The NFRs maintainability is not correlated since it has almost no values. The outcomes indicate that the mobile application developers are concerned more about these NFRs more and more with time. It hints that the software quality requirements (usability, reliability, functionality, and efficiency) are of more interest as time progresses, which recommends that the mobile apps developers, managers, and researchers should give more consideration to these NFRs.

E. RESULTS OF RQ4

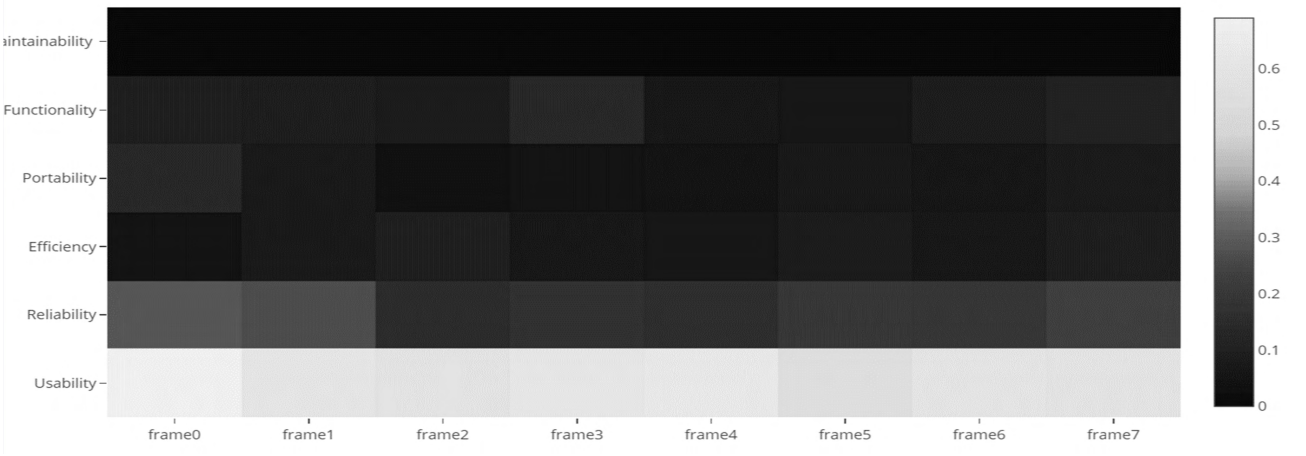
For addressing RQ4, we were primarily interested in investigating whether certain NFRs are comparatively challenging than others in the diverse specific iOS technologies or categories. Knowing the most challenging NFRs will help software engineering researchers and real-world practitioners determine the most demanding area to work on. To identify the challenges, we primarily investigated how probable it is for those problems to be successfully answered. As depicted in Table 6, we utilize four metrics to identify the difficulty of a topic from [19]. The four metrics used in our study are; ''meantime,'' ''median time,'' ''Percentage Accepted,'' and ''Average number of Answers'' as illustrated in Table 6. Usually, a question posted related to any topic or problem on SO; the questioner accepts the answer as an acceptable answer when it fulfills the desired needs of the questioner's. We examine the time it takes for a question(s) to obtain an accepted answer by deducting the creation date of the accepted answer and the question posts on SO. Then, we compute the meantime and median time of questions on SO on a particular topic. Once finding all the desired metrics, we calculate the difficulty of each of the NFRs through ensemble averaging and a linear combination via the NFRs labels of topics.

Figure 11 illustrates the difficulty of each of the NFRs in different specific iOS technologies or categories. It is evident from Figure 11 that usability and reliability followed by functionality and portability NFRs are commonly more difficult than others in the majority of the iOS technologies or categories. In the ten different iOS categories, ten out of ten are usability and reliability the most challenging NFRs. Similarly, six out of ten is the functionality, and four

a)



b)



c)

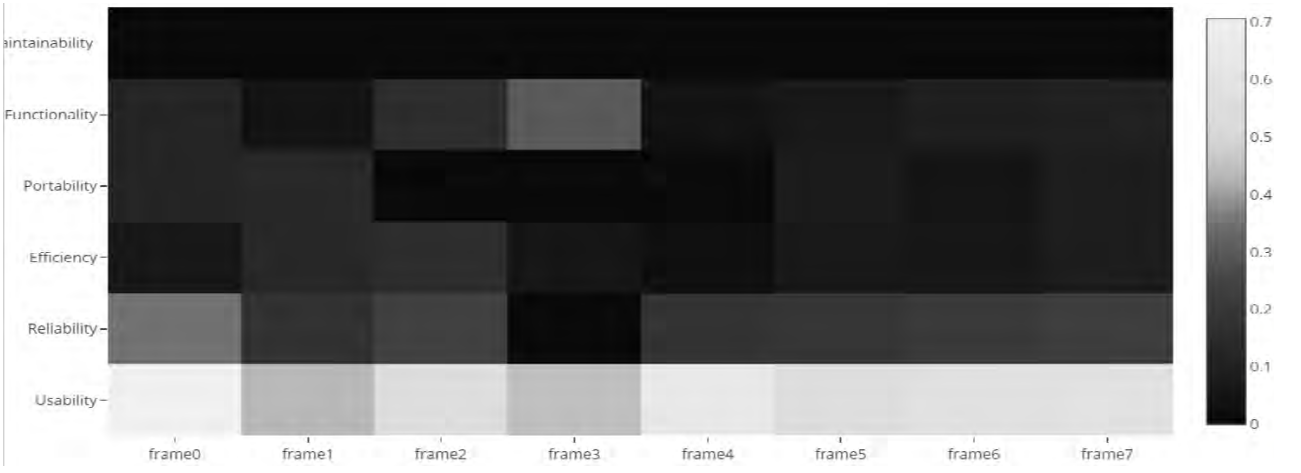
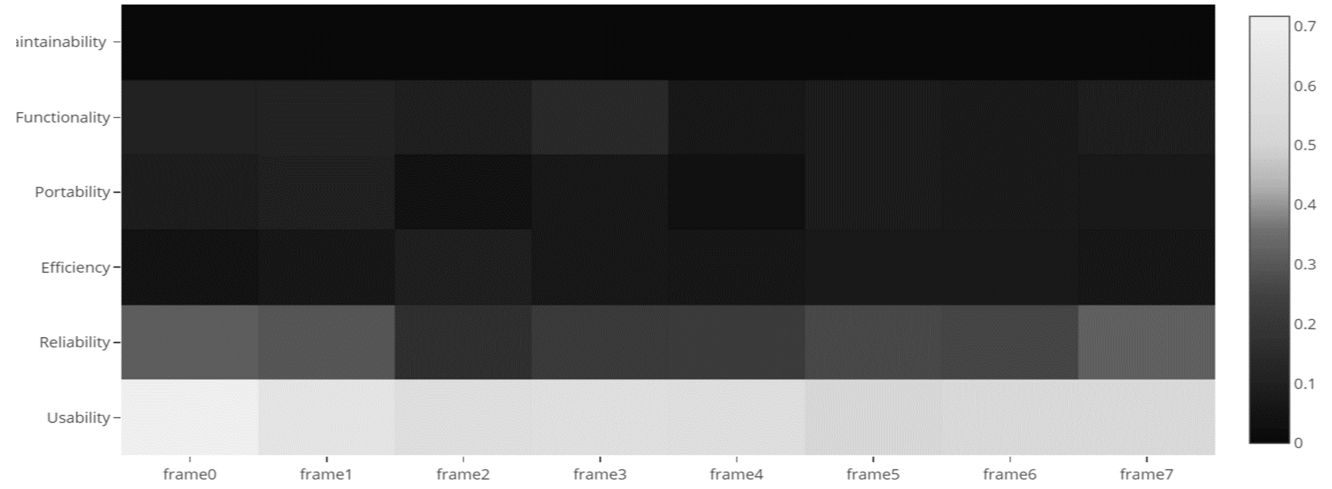
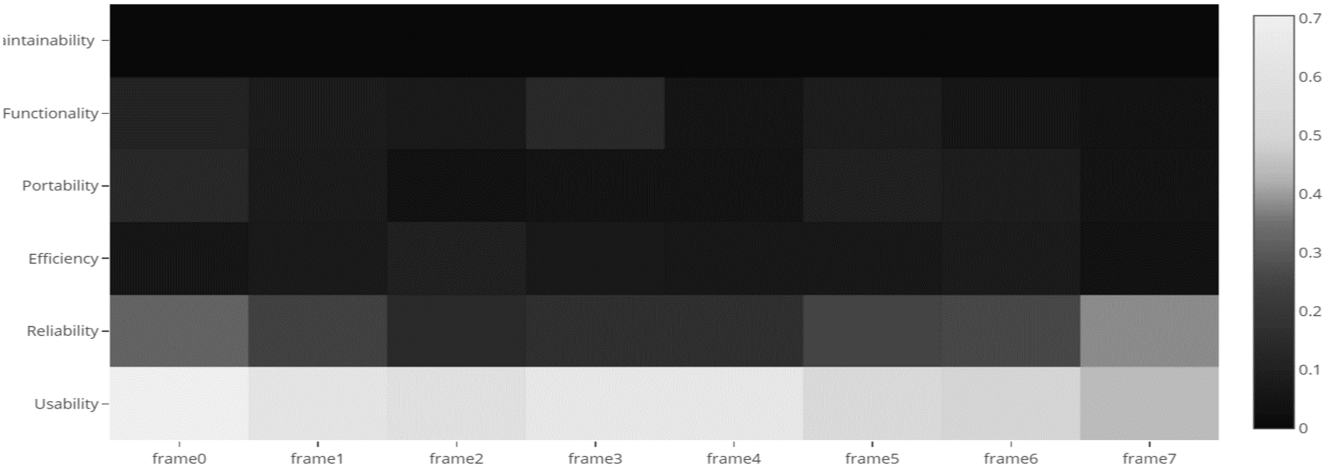


FIGURE 10. The rate of different NFRs over time in specific iOS technologies or categories. a) iPhone. b) Objective C. c) Swift. d) Xcode. e) Cocoa. f) Cordova. g) Coredata. h) iPad. i) UItableview. j) UIview.

d)



e)



f)

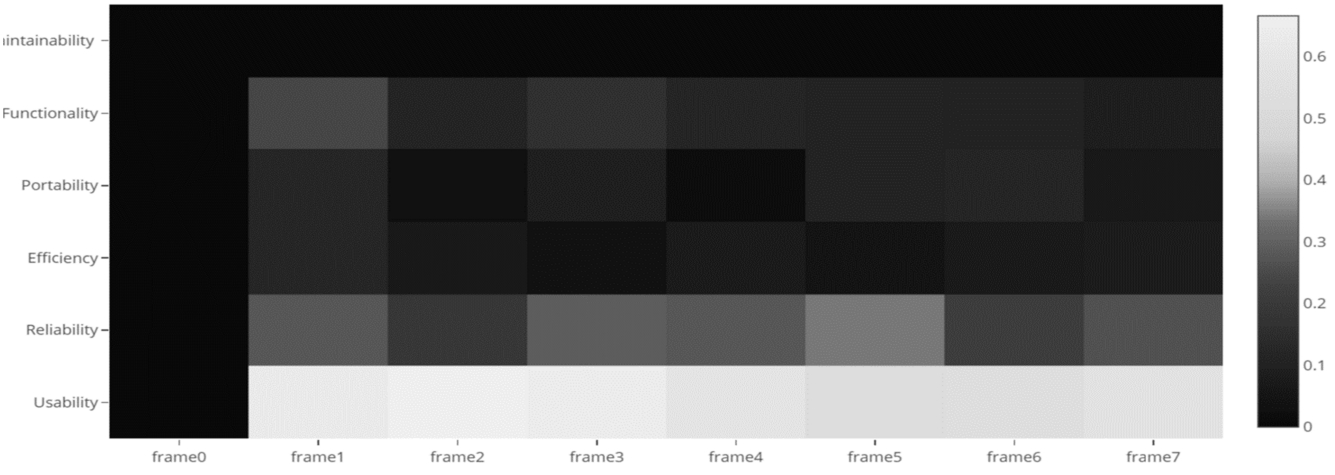


FIGURE 10. *(Continued.)* **The rate of different NFRs over time in specific iOS technologies or categories. a) iPhone. b) Objective C. c) Swift. d) Xcode. e) Cocoa. f) Cordova. g) Coredata. h) iPad. i) UItableview. j) UIview.**

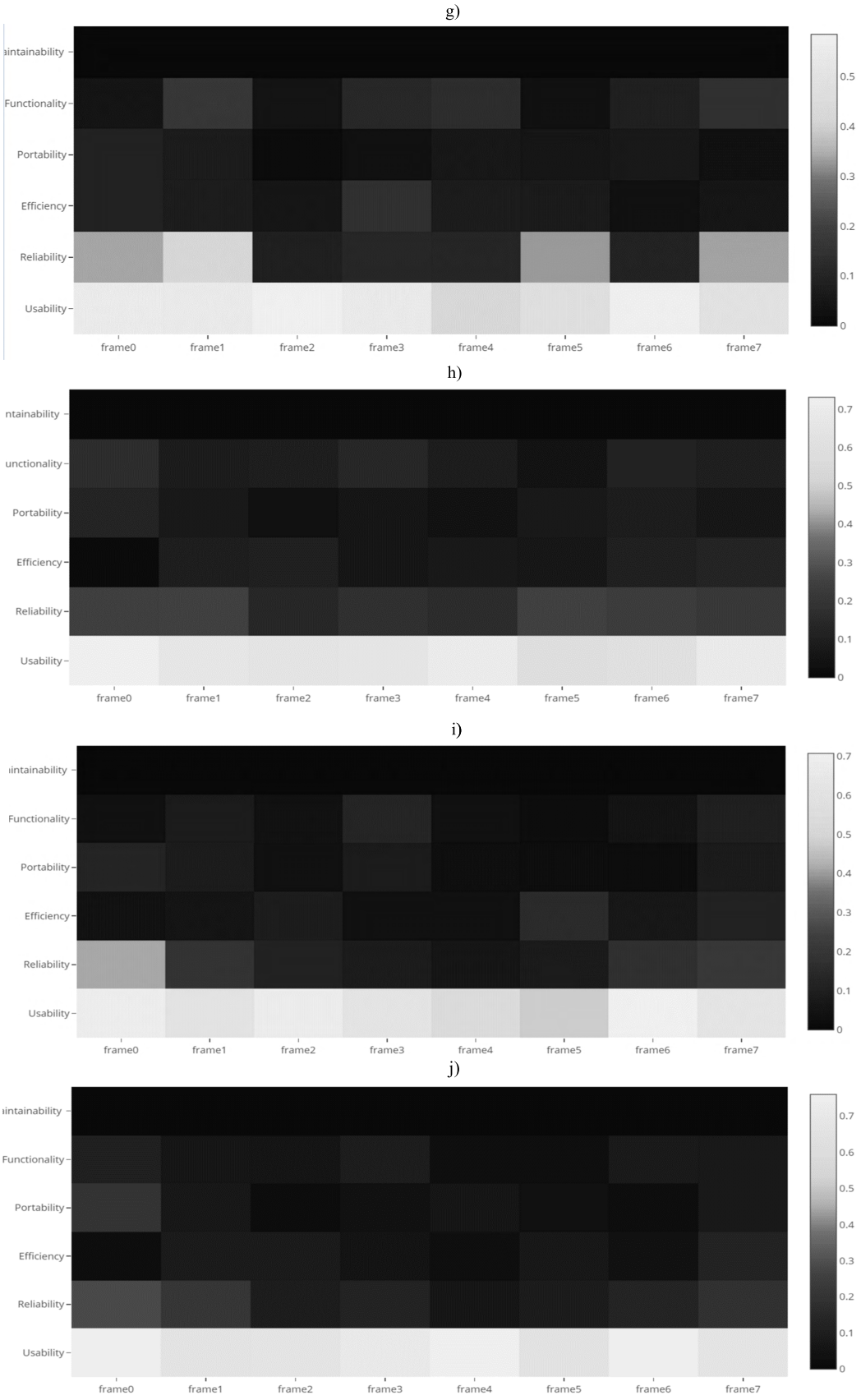


FIGURE 10. *(Continued.)* **The rate of different NFRs over time in specific iOS technologies or categories. a) iPhone. b) Objective C. c) Swift. d) Xcode. e) Cocoa. f) Cordova. g) Coredata. h) iPad. i) UItableview. j) UIview.**

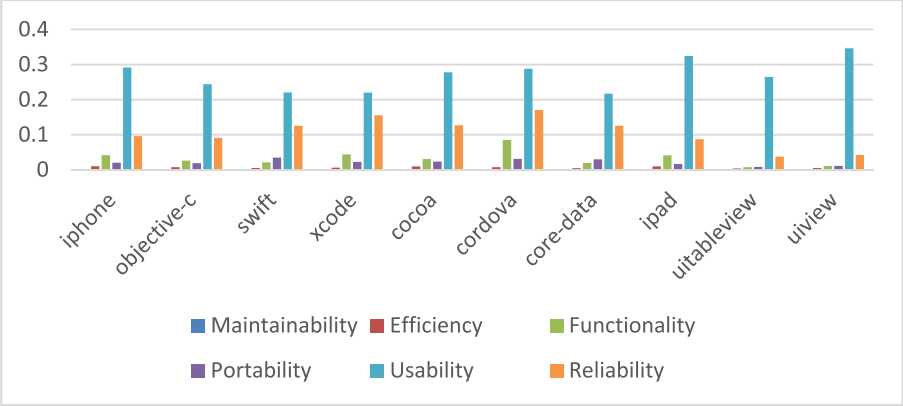


FIGURE 11. The difficulty of specific NFRs in different iOS technologies or categories.

out of ten is the portability the most difficult NFRs. The average difficulty value of usability of the ten iOS categories is the highest amongst all. It indicates that usability is the most challenging NFRs followed by reliability among all iOS technologies or categories. The SO questions related to usability had a slightly above average time to obtain accepted answers. Thus, put them in the lowest half for the average ratio of SO questions with accepted answers, and also a less average number of answers to the SO questions. Besides, it is also found that efficiency and maintainability (almost none) are the least difficult NFRs in all iOS categories. It recommends the real world application developers and researchers to invest more resources in usability and less on efficiency and maintainability. Moreover, the results from RQ1 revealed that usability is the hottest or most popular NFR and the results from RQ4 indicate that the usability turns out to be the most difficult NFRs amongst all iOS categories.

F. RESULTS OF RQ5

For addressing this research question, we explored the six different NFRs difficulty over time in the iOS specific technologies or categories. The heat map of six different NFRs difficulty over time in iOS categories is illustrated in Figure 12. The various trends of each of the six NFRs difficulty in the respective iOS category are evident from Figure 12, and it can be observed that most of the quality requirements or NFRs evolve with time. It specifies that the difficulty level evolves as time progresses. It is evident that an upward difficulty trend is observed in usability NFRs in some of the iOS categories. Nonetheless, the difficulty level of usability NFR is still high in almost all iOS categories which warrant that more emphasis should be put on it to solve the issues raised in those questions. There are also some NFRs that fluctuate as the time evolves like reliability and functionality NFRs in most of the iOS categories. It depicts that the difficulty in reliability and functionality NFRs fluctuation possibly could be sporadic and reactive to external factors. Generally, the heat map depicts that the difficulty of six different NFRs in some of the iOS categories might become more challenging as the time evolves. We also discover that the difficulties of NFRs in the starting years are considerably higher than the other years.

It possibly could be due to the proportion of the accepted iOS answers and the aggregate numbers of iOS answers are relatively fewer than the later years.

V. THREATS TO VALIDITY

Some of the potential validity threats [15], [21], [101]–[103] of our study are discussed below:

A. INTERNAL VALIDITY

For our LDA model, we selected to use 20 as the number of topics parameter (K) for each of the specified period since same words from different topics are not so frequent when the value of K = 20. However, the value of K = 20 is not necessarily the best choice but proved to be an appropriate value for NFRs analysis as reported in [104]–[106]. The motive is to look for an optimal value between the available coarser and finer grained values. To minimize this validity threat, we tested with diverse values and selected the one that provided better outcomes by looking at the identical words from diverse topics and the aggregate prevailing topic probabilities specified by the LDA model. Also, to minimize the validity threat of value of K = 20, in future, we plan and welcome other researchers to use some machine learning method for selecting the appropriate number of topics to be extracted by the LDA. Nevertheless, K = 20 is not necessarily the ideal selection, but it proved to be a suitable value for our NFRs study analysis, and after testing several values of ''K'' we are confident enough that the value of K = 20 is appropriate for our study.

The other validity threat is the reliability of the six different NFRs labeling performed. To minimize this validity threat, we assess the six different NFRs labeling on 12 months of iOS data using the measure of recall and precision rates, and the outcomes revealed it might not be precisely correct. Nonetheless, it is acceptable enough. While employing the LDA topic model, LDA models the inside contents of iOS posts as plain

TABLE 5. Spearman's coefficients results.

Categories	NFRs	Spearman's coefficients (r_s)	Categories	NFRs	Spearman's coefficients (r_s)
Cocoa	Functionality	.619	**Objective-C**	Functionality	.219
	Portability	.314		Portability	.048
	Maintainability	--		Maintainability	--
	Efficiency	.190		Efficiency	.286
	Reliability	.633		Reliability	.605
	Usability	**.786**		**Usability**	**.714**
Cordova	Functionality	.333	**Swift**	Functionality	.433
	Portability	.262		Portability	.180
	Maintainability	--		Maintainability	--
	Efficiency	.333		Efficiency	.190
	Reliability	.619		Reliability	.619
	Usability	**.714**		**Usability**	**.905**
Core-data	Functionality	.314	**UItableview**	Functionality	.333
	Portability	.333		**Portability**	**.714**
	Maintainability	--		Maintainability	--
	Efficiency	**.714**		Efficiency	.476
	Reliability	.310		Reliability	.238
	Usability	.333		Usability	.619
iPad	Functionality	.333	**UIview**	Functionality	.362
	Portability	.143		Portability	.238
	Maintainability	--		Maintainability	--
	Efficiency	.524		Efficiency	.310
	Reliability	.614		Reliability	.457
	Usability	**.786**		Usability	.619
iPhone	Functionality	.333	**XCode**	Functionality	.619
	Portability	.148		Portability	.310
	Maintainability	--		Maintainability	--
	Efficiency	.357		Efficiency	.214
	Reliability	.357		Reliability	.345
	Usability	**.786**		**Usability**	**.905**

TABLE 6. The used metrics with an explanation.

Metrics	Explanation
Mean Time	Meantime is the questions posted on SO takes to have an accepted answer
Median Time	Median time is the questions posted on SO takes to have an accepted answer
% Accepted	It is the percentage of SO questions that obtained accepted answers
Avg. # of Answers	It is the average number of answers on SO the questions in a certain topic obtain

text, comprising different source codes. As there might exist a condition that iOS questions are containing more source code than the plain text, the source code is comprised of different words (i.e., classes, methods names, and variables) concerning the application area that possibly could lead LDA model to classify the concepts incorrectly. Nevertheless, in comparison with the number of words in total, there is a trivial number of application area related words. The investigation offers some form of generalization. We leave this as future work to conduct an improved investigation by taking into consideration source code removal from SO posts.

For questions classification to the different iOS technologies or categories, we primarily used the tags of the iOS posts. The possibility exists that the tags of iOS posts are left by the mobile application developers or incorrectly labeled. To minimize this threat, we performed manual inspection of the iOS posts related to the different technologies or categories and verified that they possess the suitable iOS questions. The other threat is related to the selection of ten iOS technologies or categories, maybe there are other more comprehensive categorizations which contain all the discussions in Stack Overflow, but the categorization in the paper is reasonably

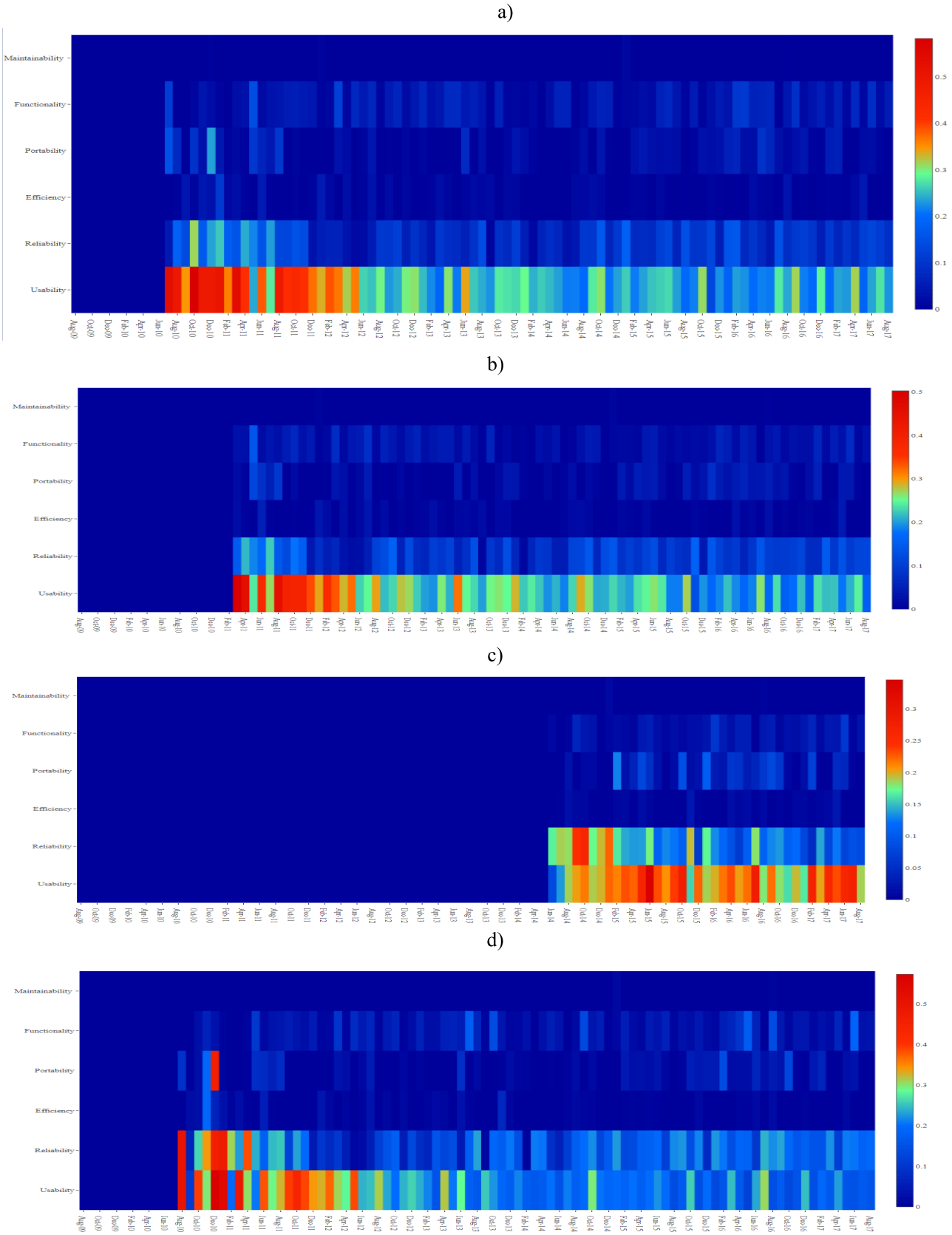


FIGURE 12. The difficulty level of six NFRs over time in iOS technologies or categories. a) iPhone. b) Objective C. c) Swift. d) Xcode. e) Cocoa. f) Cordova. g) Coredata. h) iPad. i) Ultableview. j) UIview.

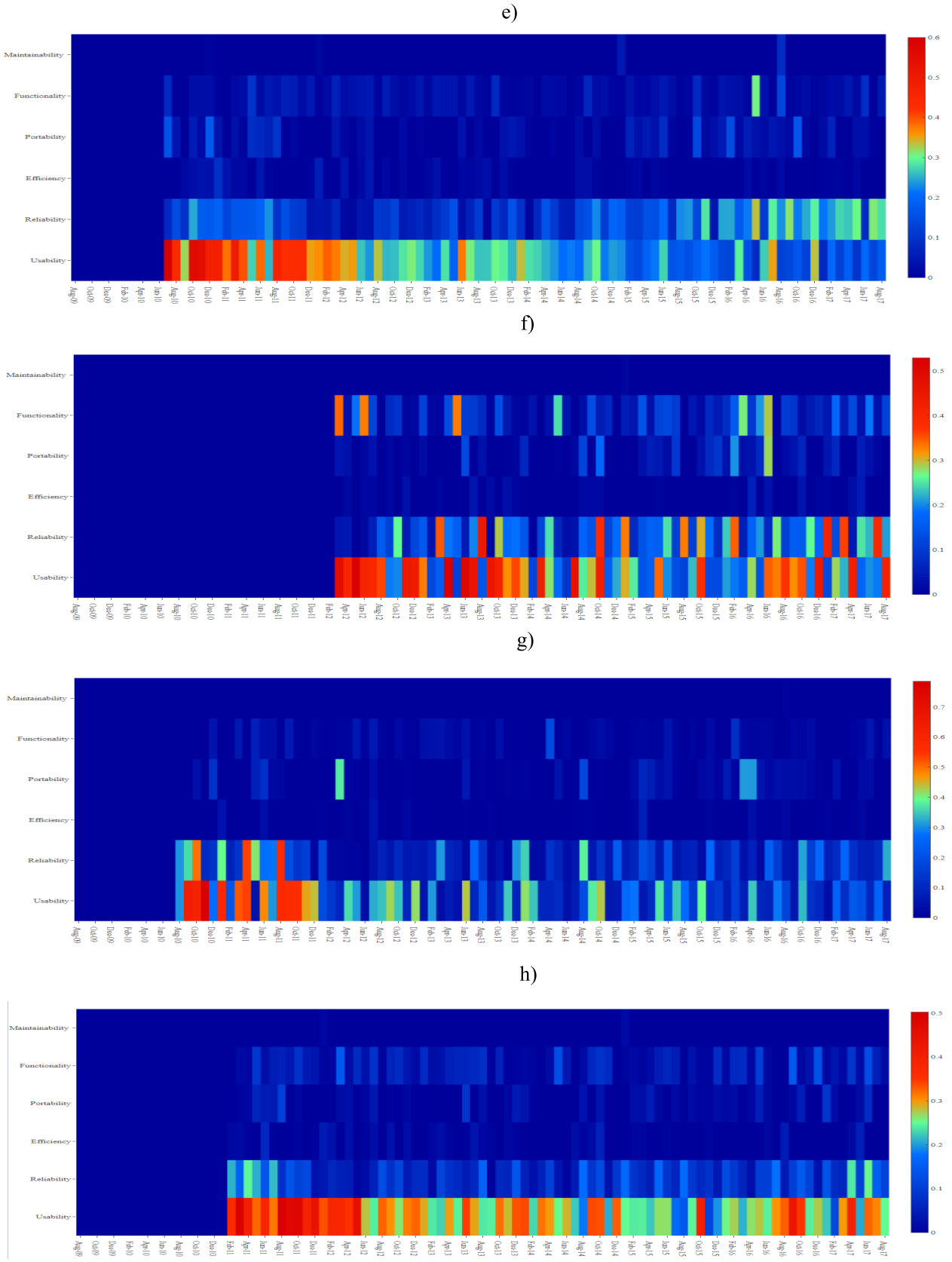


FIGURE 12. *(Continued.)* **The difficulty level of six NFRs over time in iOS technologies or categories. a) iPhone. b) Objective C. c) Swift. d) Xcode. e) Cocoa. f) Cordova. g) Coredata. h) iPad. i) UItableview. j) UIview.**

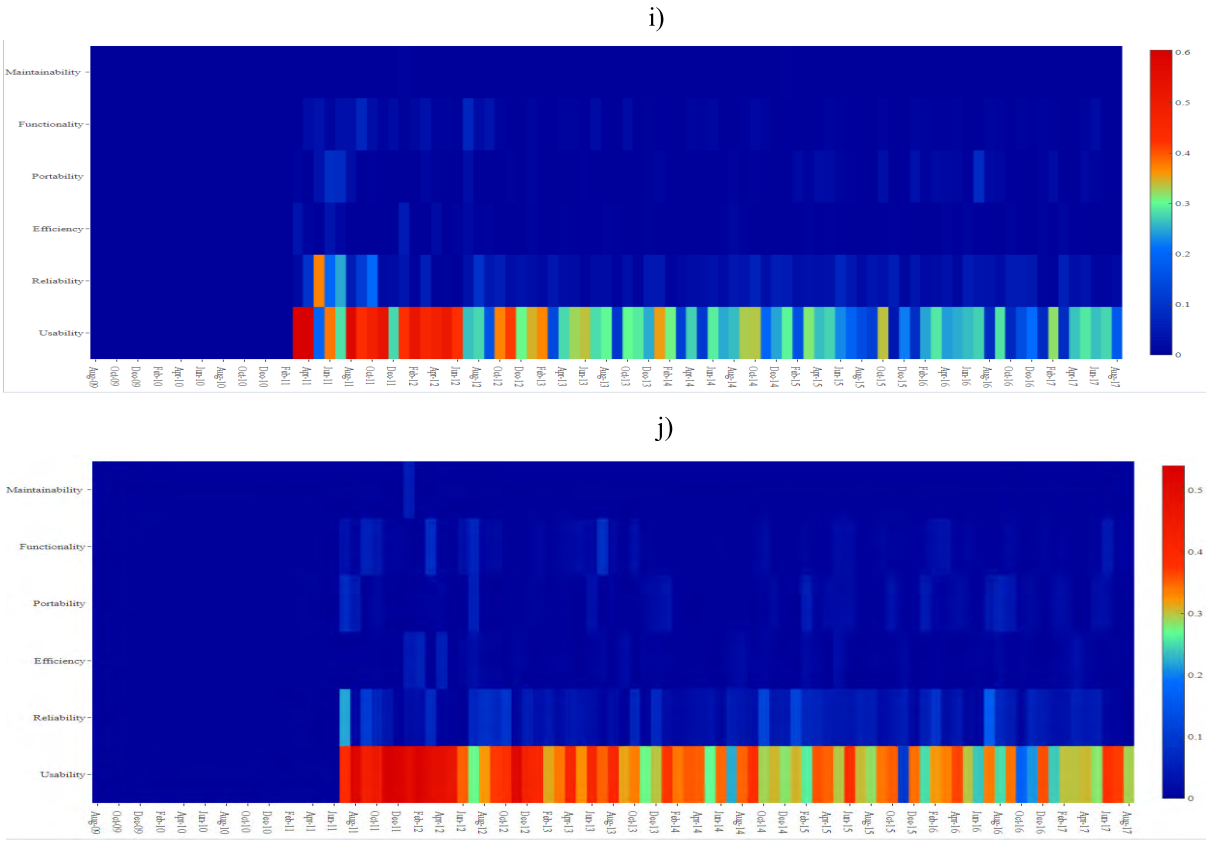


FIGURE 12. ***(Continued.)*** **The difficulty level of six NFRs over time in iOS technologies or categories. a) iPhone. b) Objective C. c) Swift. d) Xcode. e) Cocoa. f) Cordova. g) Coredata. h) iPad. i) UItableview. j) UIview.**

diverse and representative. We specifically considered Stack Overflow questions of different types to determine the NFRs in different iOS technologies or categories. Thus, our selection is quite comprehensive as it covers different iOS technologies sufficiently.

While detecting if the iOS questions were effectively answered in RQ4, we supposed that the question poser would consider the answer as accepted if it resolves their issue. Nonetheless, they are not required to perform this, and it is quite likely that we could not catch all of the successfully answered iOS questions precisely. We divided the whole corpus into certain time-windows by using the period size equal to 30 days like [31]. We lack the proof to specify that selecting 30 days period size is the appropriate distribution, though it is a considerably suitable medium granularity value for our six different NFRs analysis. The period comprising excessively many posts and thus accumulating excessively many words might cause the discrepancy between topics is not apparent sufficiently, while the period comprising scarce data is not enough for conducting the analysis. The 30 days size is lesser than the period having excessively many posts but considerably substantial for a sufficient number of posts to examine. Besides, some other internal threats to the validity of this study were minimized by utilizing certain available developed tools for extracting the desired iOS posts data, executing the LDA model, labeling topics with six different NFRs and finally visualizing the results. We used the Python library Beautiful Soup to extract and parse iOS posts data, Python sklearn to build our LDA topic model, Python for labeling the six different NFRs, Microsoft Excel and plotly [107] for visualizing the figures.

B. EXTERNAL VALIDITY

One of the possible threats of this study is that we only used Stack Overflow posts data. Nevertheless, SO is one of the well-known and widely used programming Q&A website by software developers across the globe. SO is populated with thousands of diverse programming related questions daily. In this study, we used the data dump from 31st July 2008 up to 31st August 2017. The scale and diversity of SO give some form of generalization. In the future, we plan to conduct and welcome other researchers to do more enhanced analysis by adding other similar programming Q&A sites and forums. Besides, applying ISO9126 is also a threat to the validity as there is lack of evidence to state that ISO9126 is the only suitable standard. However, ISO9126 currently is the most commonly used software quality model and delivers a universally recognized terminology for software quality in the software industry [35], [108]. Therefore, we consider it is adequate to use it for this study. In the future, we would

take into consideration other commonly practiced software quality models, for instance, ISO25010.

C. CONSTRUCT VALIDITY

The reason behind selecting SO as a Q&A data set is due to the presence of a diverse set of developers who post questions and answers related to the iOS mobile application development. There is a possibility that some of the posts could be posted by the end user which could be a threat to the validity of the results. Nevertheless, the authors are confident that the ratio of such posts are quite less as the majority of the questions posted on Stack Overflow are by professional programmers. Thus, the authors can confidently conclude that the experiment iOS posts data was by in large discussed by experts iOS programmers on SO and thus the results are considered reliable. Moreover, to reduce this validity threat, the authors plan and welcome other researchers to investigate only those iOS posts which have gained more votes and answers.

D. CONCLUSION VALIDITY

While building the validation corpus of this study, the annotators might have some level of biases due to the difference in understanding and subjectivity. To reduce the effect of this threat to minimal, we invited four Ph.D. students in Software Engineering to do the labeling task of the topics manually. Besides, we also assessed the inter-rater reliability agreements between the annotators labeling task using Kendall's coefficient of concordance (W) [97].

VI. CONCLUSION AND FUTURE WORK

We used LDA topic model to identify and evaluate the different NFRs discussed in iOS application development on Stack Overflow posts. This study offers an approximate comprehension of the different NFRs through the contemporary diverse iOS practitioners' eyes. It highlighted and recommended the real problems and needs of the diverse iOS application developers. We have primarily investigated the different NFRs the iOS developers concentrated on, the NFRs problems or difficulties the iOS developers confronted with, and the evolving trends of different NFRs difficulty and focus with the evolution of time. Our findings revealed that iOS developers focus mostly on usability, reliability, and functionality. They are found comparatively to be less focused on efficiency and portability, while maintainability is almost neglected. The outcomes of using posts alone in comparison with the output of using posts along with the comments yielded similar results. The most problematic areas left unresolved lies in usability, reliability, and functionality, which hints of more work to do in the future in these areas to improve iOS development.

We also examine the trends of the six quality requirements or NFRs on the iOS posts and the unanswered iOS questions. The trend analysis of the six different quality requirements or NFRs yielded that they change over time. The evolution of NFRs like reliability, portability, and functionality will raise the attention of the iOS developers, and the usability will most probably stay hot in the coming years. Besides, we explore the six NFRs in specific iOS technologies or categories and examine the associations and discrepancies amongst them. The outcome revealed that the quality NFR is considered as a similar concern amongst diverse iOS technologies or categories as the iOS developers' emphasis more on the usability NFR, and less on the maintainability NFR. The results also depicted that some of the NFRs are of more interest amongst iOS developers as the time evolves. Furthermore, we examine the difficulty of the six NFRs for each of the iOS categories by using the four metrics. We find that usability is the most challenging NFR, whereas efficiency and maintainability are the least challenging NFRs faced by iOS developers. The outcomes from examining the difficulty trend of NFRs with the evolution of time in the ten iOS categories depicted that most of the NFRs difficulties changes over time, and an upward difficulty trend is observed in usability NFRs in some of the iOS categories. Moreover, all these findings suggest that the content shared on Stack Overflow posts should be considered more thoroughly and thoughtfully as an elicitation source for NFRs or quality requirements. In the future, we welcome other researchers and would like to focus more intensely on specific iOS development technologies to analyze the needs and problems of iOS developers.

REFERENCES

[1] M. A. Alnuem, A. Ahmad, and H. Khan, "Requirements understanding: A challenge in global software development, industrial surveys in kingdom of Saudi Arabia," in *Proc. IEEE 36th Annu. Comput. Softw. Appl. Conf.*, Izmir, Turkey, Jul. 2012, pp. 297–306.

[2] H. Khan, A. Ahmad, C. Johansson, and M. A. Al Nuem, "Requirements understanding in global software engineering: Industrial surveys," in *Proc. Int. Conf. Comput. Softw. Modeling (IPCSIT)*, 2011, pp. 167–173.

[3] H. Khan, A. Ahmad, and M. A. Alnuem, "Knowledge management: A solution to requirements understanding in global software engineering," *Res. J. Appl. Sci., Eng. Technol.*, vol. 7, no. 14, pp. 2087–2099, 2012.

[4] A. Ahmad and H. Khan, "The importance of knowledge management practices in overcoming the global software engineering challenges in requirements understanding," M.S. thesis, Blekinge Inst. Technol., Karlskrona, Sweden, 2008.

[5] K. Wnuk, D. Pfahl, D. Callele, and E.-A. Karlsson, "How can open source software development help requirements management gain the potential of open innovation: An exploratory study," in *Proc. ESEM*, 2012, pp. 271–279.

[6] M. Kauppinen, J. Savolainen, and T. Mannisto, "Requirements engineering as a driver for innovations," in *Proc. 15th IEEE Int. Requirements Eng. Conf.*, Oct. 2007, pp. 15–20.

[7] H. Munir, P. Runeson, and K. Wnuk, "A theory of openness for software engineering tools in software organizations," *Inf. Softw. Technol.*, vol. 97, pp. 26–45, May 2018.

[8] J. Linåker, B. Regnell, and H. Munir, "Requirements engineering in open innovation: A research agenda," in *Proc. ICSSP*, Tallinn, Estonia, 2015, pp. 208–212.

[9] H. Munir, K. Wnuk, and P. Runeson, "Open innovation in software engineering: A systematic mapping study," *Empirical Softw. Eng.*, vol. 21, pp. 684–723, 2016.

[10] H. W. Chesbrough, *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Boston, MA, USA: Harvard Business School Press, 2003.

[11] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?: Nier track," in *Proc. 33rd Int. Conf. Softw. Eng. (ICSE)*, May 2011, pp. 804–807.

[12] E. C. Groen, S. Kopczyńska, M. P. Hauer, T. D. Krafft, and J. Doerr, "Users—The hidden software product quality experts?: A study on how app users report quality aspects in online reviews," in *Proc. IEEE 25th Int. Requirements Eng. Conf.*, Sep. 2017, pp. 80–89.

[13] J. Vassileva, "Toward social learning environments," *IEEE Trans. Learn. Technol.*, vol. 1, no. 4, pp. 199–214, Oct. 2008.

[14] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Proc. IEEE 22nd Int. Requirements Eng. Conf.*, Aug. 2014, pp. 153–162.

[15] A. Ahmad, C. Feng, S. Ge, and A. Yousif, "A survey on mining stack overflow: Question and answering (Q&A) community," *Data Technol. Appl.*, vol. 52, 2, pp. 190–247, 2018.

[16] D. Ameller, C. Ayala, J. Cabot, and X. Franch, "Non-functional requirements in architectural decision making," *IEEE Softw.*, vol. 30, no. 2, pp. 61–67, Mar. 2013.

[17] L. Chung and J. C. S. do Prado Leite, "On non-functional requirements in software engineering," in *Conceptual Modeling: Foundations and Applications*, 2009.

[18] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, "Non-functional requirements in industry-three case studies adopting an experience-based NFR method," in *Proc. 13th IEEE Int. Conf. Requirements Eng.*, Aug./Sep. 2005, pp. 373–382.

[19] C. Rosen and E. Shihab, "What are mobile developers asking about? A large scale study using stack overflow," *Empirical Softw. Eng.*, vol. 21, pp. 1192–1223, Jun. 2016.

[20] A. Ahmad, C. Feng, M. Tao, A. Yousif, and S. Ge, "Challenges of mobile applications development: Initial results," in *Proc. 8th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Beijing, China, Nov. 2017, pp. 464–469.

[21] A. Ahmad, K. Li, C. Feng, S. M. Asim, A. Yousif, and S. Ge, "An empirical study of investigating mobile applications development challenges," *IEEE Access*, vol. 6, pp. 17711–17728, 2018.

[22] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development issues using stack overflow," in *Proc. 10th Work. Conf. Mining Softw. Repositories*, May 2013, pp. 93–96.

[23] M. Rebouças, G. Pinto, F. Ebert, W. Torres, A. Serebrenik, and F. Castor, "An empirical study on the usage of the swift programming language," in *Proc. IEEE 23rd Int. Conf. Softw. Anal., Evol., Reeng. (SANER)*, Mar. 2016, pp. 634–638.

[24] I. K. Villanes, S. M. Ascate, J. Gomes, and A. C. Dias-Neto, "What are software engineers asking about android testing on stack overflow?" in *Proc. SBES*, Rio de Janeiro, Brazil, 2017, pp. 104–113.

[25] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in stack overflow," *Empirical Softw. Eng.*, vol. 19, no. 3, pp. 619–654, 2014.

[26] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, and X. Zhang, "Which non-functional requirements do developers focus on? An empirical study on stack overflow using topic analysis," in *Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories*, May 2015, pp. 446–449.

[27] G. Pinto, W. Torres, and F. Castor, "A study on the most popular questions about concurrent programming," in *Proc. 6th Workshop Eval. Usability Program. Lang. Tools*, 2015, pp. 39–46.

[28] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *Proc. 11th Work. Conf. Mining Softw. Repositories*, 2014, pp. 112–121.

[29] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[30] Ø. Hauge, C. Ayala, and R. Conradi, "Adoption of open source software in software-intensive organizations—A systematic literature review," *Inf. Softw. Technol.*, vol. 52, pp. 1133–1154, Nov. 2010.

[31] A. Hindle, M. W. Godfrey, and R. C. Holt, "What's hot and what's not: Windowed developer topic analysis," in *Proc. IEEE Int. Conf. Softw. Maintenance*, Sep. 2009, pp. 339–348.

[32] A. Hindle, N. Ernst, M. W. Godfrey, R. C. Holt, and J. Mylopoulos, "What's in a name? On the automated topic naming of software maintenance activities," Tech. Rep., 2010.

[33] A. Hindle, C. Bird, T. Zimmermann, and N. Nagapan, "Relating requirements to implementation via topic analysis: Do topics extracted from requirements make sense to managers and developers?" in *Proc. 28th IEEE Int. Conf. Softw. Maintenance (ICSM)*, Trento, Italy, Sep. 2012, pp. 243–252.

[34] A. Hindle, N. A. Ernst, M. W. Godfrey, and J. Mylopoulos, "Automated topic naming to support cross-project analysis of software maintenance activities," in *Proc. 8th Work. Conf. Mining Softw. Repositories*, 2011, pp. 163–172.

[35] "9126 Software product evaluation-quality characteristics and guidelines for their use," Int. Standard Org., Geneva, Switzerland, Tech. Rep., 2001.

[36] A. Ahmad, L. Kan, C. Feng, and T. Sun, "An empirical study on how iOS developers report quality aspects on stack overflow," *Int. J. Mach. Learn. Comput. (IJMLC)*, vol. 8, pp. 501–506, Oct. 2018.

[37] 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," *J. Comput. Sci. Technol.*, vol. 31, pp. 910–924, Sep. 2016.

[38] G. H. Pinto and F. Kamei, "What programmers say about refactoring tools?: An empirical investigation of stack overflow," in *Proc. ACM Workshop Workshop Refactoring Tools*, 2013, pp. 33–36.

[39] A. Fontão *et al.*, "Supporting governance of mobile application developers from mining and analyzing technical questions in stack overflow," *J. Softw. Eng. Res. Develop.*, vol. 6, p. 8, Aug. 2018.

[40] M. Martinez and S. Lecomte. (2017). "Discovering discussion topics about development of cross-platform mobile applications using a cross-compiler development framework."

[41] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest Q&A site in the west," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2011, pp. 2857–2866.

[42] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider, "Answering questions about unanswered questions of stack overflow," in *Proc. 10th Work. Conf. Mining Softw. Repositories*, 2013, pp. 97–100.

[43] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft, "Building reputation in stackoverflow: An empirical investigation," in *Proc. 10th Work. Conf. Mining Softw. Repositories*, May 2013, pp. 89–92.

[44] B. Bazelli, A. Hindle, and E. Stroulia, "On the Personality Traits of StackOverflow Users," in *Proc. ICSM*, Sep. 2013, pp. 460–463.

[45] Y. Jin, X. Yang, R. G. Kula, E. Choi, K. Inoue, and H. Iida, "Quick trigger on stack overflow: A study of gamification-influenced member tendencies," in *Proc. 12th Work. Conf. Mining Softw. Repositories*, 2015, pp. 434–437.

[46] J. Goderie, B. M. Georgsson, B. Van Graafeiland, and A. Bacchelli, "Eta: Estimated time of answer predicting response time in Stack Overflow," in *Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories (MSR)*, 2015, pp. 414–417.

[47] F. Calefato, F. Lanubile, M. C. Marasciulo, and N. Novielli, "Mining successful answers in stack overflow," in *Proc. 12th Work. Conf. Mining Softw. Repositories*, 2015, pp. 430–433.

[48] N. Novielli, F. Calefato, and F. Lanubile, "The challenges of sentiment detection in the social programmer ecosystem," in *Proc. 7th Int. Workshop Social Softw. Eng.*, 2015, pp. 33–40.

[49] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Discovering value from community activity on focused question answering sites: A case study of stack overflow," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 850–858.

[50] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Steering user behavior with badges," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 95–106.

[51] S. Grant and B. Betts, "Encouraging user behaviour with achievements: An empirical study," in *Proc. 10th Work. Conf. Mining Softw. Repositories (MSR)*, 2013, pp. 65–68.

[52] A. Marder, "Stack overflow badges and user behavior: An econometric approach," in *Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories*, May 2015, pp. 450–453.

[53] R. Slag, M. de Waard, and A. Bacchelli, "One-day flies on stackoverflow-why the vast majority of stackoverflow users only posts once," in *Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories*, May 2015, pp. 458–461.

[54] 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 *Proc. 28th IEEE Int. Conf. Softw. Maintenance (ICSM)*, Sep. 2012, pp. 25–34.

[55] V. Honsel, S. Herbold, and J. Grabowski, "Intuition vs. truth: Evaluation of common myths about stackoverflow posts," in *Proc. 12th Work. Conf. Mining Softw. Repositories (MSR)*, 2015, pp. 438–441.

[56] S. A. Chowdhury and A. Hindle, "Mining StackOverflow to filter out off-topic IRC discussion," in *Proc. 12th Work. Conf. Mining Softw. Repositories*, 2015, pp. 422–425.

[57] H. Li, Z. Xing, X. Peng, and W. Zhao, "What help do developers seek, when and how?" in *Proc. 20th Work. Conf. Reverse Eng. (WCRE)*, Oct. 2013, pp. 142–151.
[58] 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 Inst. Technol., Atlanta, GA, USA, Tech. Rep., 2012.
[59] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE Trans. Softw. Eng.*, vol. 43, no. 9, pp. 817–847, Sep. 2017.
[60] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: MSR for app stores," in *Proc. 9th IEEE Work. Conf. Mining Softw. Repositories*, 2012, pp. 108–111.
[61] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proc. IEEE Mining Softw. Repositories (MSR)*, San Francisco, CA, USA, May 2013, pp. 41–44.
[62] L. V. G. Carreno and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *Proc. IEEE Int. Conf. Softw. Eng. (ICSE)*, May 2013, pp. 582–591.
[63] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *Proc. 21st IEEE Int. Requirements Eng. Conf. (RE)*, Rio de Janeiro, Brazil, Jul. 2013, pp. 125–134.
[64] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Eng.*, vol. 21, pp. 311–331, Sep. 2016.
[65] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Proc. IEEE 23rd Int. Requirements Eng. Conf. (RE)*, Ottawa, ON, Canada, Aug. 2015, pp. 116–125.
[66] H. Li, L. Zhang, L. Zhang, and J. Shen, "A user satisfaction analysis approach for software evolution," in *Proc. IEEE Int. Conf. Prog. Inform. Comput. (PIC)*, Dec. 2010, pp. 1093–1097.
[67] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proc. Int. Conf. Knowl. Discovery Data Mining (KDD)*, Aug. 2013, pp. 1276–1284.
[68] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?" *IEEE Softw.*, vol. 32, no. 3, pp. 70–77, May 2015.
[69] G. Bavota, M. Linares-Vásquez, C. E. Bernal-Cárdenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "The impact of API change-and fault-proneness on the user ratings of android apps," *IEEE Trans. Softw. Eng.*, vol. 41, no. 4, pp. 384–407, Apr. 2015.
[70] L. Chung and B. A. Nixon, "Dealing with non-functional requirements: Three experimental studies of a process-oriented approach," in *Proc. 17th Int. Conf. Softw. Eng.*, Apr. 1995, pp. 25–37.
[71] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using non-functional requirements: A process-oriented approach," *IEEE Trans. Softw. Eng.*, vol. 18, no. 6, pp. 483–497, Jun. 1992.
[72] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Springer, 2000.
[73] D. Mairiza, D. Zowghi, and N. Nurmuliani, "An investigation into the notion of non-functional requirements," in *Proc. ACM Symp. Appl. Comput.*, 2010, pp. 311–317.
[74] M. Glinz, "On non-functional requirements," in *Proc. 15th IEEE Int. Requirements Eng. Conf.*, Oct. 2007, pp. 21–26.
[75] M. Umar and N. A. Khan, "Analyzing non-functional requirements (NFRs) for software development," in *Proc. IEEE 2nd Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Jul. 2011, pp. 675–678.
[76] M. Galster and E. Bucherer, "A taxonomy for identifying and specifying non-functional requirements in service-oriented development," in *Proc. IEEE Congr. Services-I*, Jul. 2008, pp. 345–352.
[77] D. Ameller, C. Ayala, J. Cabot, and X. Franch, "How do software architects consider non-functional requirements: An exploratory study," in *Proc. 20th IEEE Int. Requirements Eng. Conf. (RE)*, Sep. 2012, pp. 41–50.
[78] D. Ameller, X. Franch, and J. Cabot, "Dealing with non-functional requirements in model-driven development," in *Proc. 18th IEEE Int. Requirements Eng. Conf. (RE)*, Sep./Oct. 2010, pp. 189–198.
[79] S. Kugele, W. Haberl, M. Tautschnig, and M. Wechs, "Optimizing automatic deployment using non-functional requirement annotations," in *Leveraging Applications of Formal Methods, Verification and Validation*, 2008, pp. 400–414.
[80] M. Ahmad, N. Belloir, and J.-M. Bruel, "Modeling and verification of functional and non-functional requirements of ambient self-adaptive systems," *J. Syst. Softw.*, vol. 107, pp. 50–70, Sep. 2015.
[81] A. Borg, A. Yong, P. Carlshamre, and K. Sandahl, "The bad conscience of requirements engineering: An investigation in real-world treatment of non-functional requirements," Tech. Rep., 2003.
[82] W. Damm, A. Votintseva, A. Metzner, B. Josko, T. Peikenkamp, and E. Böde, "Boosting re-use of embedded automotive applications through rich components," in *Foundations of Interface Technologies*, 2005.
[83] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "The detection and classification of non-functional requirements with application to early aspects," in *Proc. 14th IEEE Int. Requirements Eng. Conf.*, Sep. 2006, pp. 39–48.
[84] J. Eckhardt, D. M. Fernandez, and A. Vogelsang, "How to specify non-functional requirements to support seamless modeling? A study design and preliminary results," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Oct. 2015, pp. 1–4.
[85] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are 'non-functional' requirements really non-functional? An investigation of non-functional requirements in practice," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. (ICSE)*, May 2016, pp. 832–842.
[86] E. Noei, M. D. Syer, Y. Zou, A. E. Hassan, and I. Keivanloo, "A study of the relation of mobile device attributes with the user-perceived quality of Android apps," *Empirical Softw. Eng.*, vol. 22, pp. 3088–3116, Dec. 2017.
[87] M. Yan, X. Zhang, D. Yang, L. Xu, and J. D. Kymer, "A component recommender for bug reports using discriminative probability latent semantic analysis," *Inf. Softw. Technol.*, vol. 73, pp. 37–51, May 2016.
[88] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn, "Bug localization using latent dirichlet allocation," *Inf. Softw. Technol.*, vol. 52, pp. 972–990, Sep. 2010.
[89] J. Zou, L. Xu, M. Yan, X. Zhang, J. Zeng, and S. Hirokawa, "Automated duplicate bug report detection using multi-factor analysis," *IEICE Trans. Inf. Syst.*, vol. E99.D, no. 7, pp. 1762–1775, 2016.
[90] M. Allamanis and C. Sutton, "Why, when, and what: Analyzing stack overflow questions by topic, type, and code," in *Proc. 10th Work. Conf. Mining Softw. Repositories*, May 2013, pp. 53–56.
[91] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[92] Y. Fu, M. Yan, X. Zhang, L. Xu, D. Yang, and J. D. Kymer, "Automated classification of software change messages by semi-supervised Latent Dirichlet Allocation," *Inf. Softw. Technol.*, vol. 57, pp. 369–377, Jan. 2015.
[93] T. T. Nguyen, T. N. Nguyen, and T. M. Phuong, "Topic-based defect prediction (NIER track)," in *Proc. 33rd Int. Conf. Softw. Eng. (ICSE)*, Honolulu, HI, USA, 2011, pp. 932–935.
[94] S. Thomas, "Mining unstructured software repositories using ir models," Ph.D. dissertation, Queen's Univ., Kingston, ON, Canada, 2012.
[95] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative evaluation of software quality," in *Proc. 2nd Int. Conf. Softw. Eng.*, 1976, pp. 592–605.
[96] J. A. McCall, "Factors in software quality," *Preliminary Handbook on Software Quality for an Acquisiton Manager*. Boston, MA, USA: General Electric, 1977.
[97] A. Eye and E.-Y. Mun, *Analyzing Rater Agreement: Manifest Variable Methods*. London, U.K.: Taylor & Francis, 2006.
[98] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.
[99] M. M. Mukaka, "Statistics corner: A guide to appropriate use of Correlation coefficient in medical research," *Malawi Med. J.*, vol. 24, no. 3, pp. 69–71, 2012.
[100] D. Hinkle, W. Wiersma, and S. G. Jurs, *Applied Statistics for the Behavioral Sciences*, 5th ed. Boston, MA, USA: Houghton Mifflin, 2003.
[101] H. K. Wright, M. Kim, and D. E. Perry, "Validity concerns in software engineering research," in *Proc. FSE/SDP Workshop Future Softw. Eng. Res. (FoSER)*, Santa Fe, NM, USA, 2010, pp. 411–414.
[102] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research-an initial survey," in *Proc. Int. Conf. Softw. Eng. Knowl. Eng.*, 2010, pp. 374–379.
[103] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering*. London, U.K.: Kluwer, 2000.

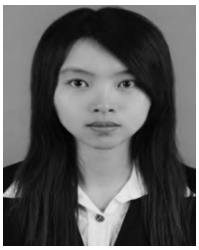
[104] S. Grant and J. R. Cordy, "Estimating the optimal number of latent concepts in source code analysis," in *Proc. 10th IEEE Working Conf. Source Code Anal. Manipulation (SCAM)*, Timisoara, Romania, Sep. 2010, pp. 65–74.
[105] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein, "Studying software evolution using topic models," *Sci. Comput. Program.*, vol. 80, pp. 457–479, Feb. 2014.
[106] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno, "Evaluation methods for topic models," in *Proc. 26th Int. Conf. Mach. Learn. (ICML)*, 2009, pp. 1105–1112.
[107] G. Maker. [Online]. Available: https://plot.ly/create/
[108] J. Bøegh, "A new standard for quality requirements," *IEEE Softw.*, vol. 25, no. 2, pp. 57–63, Mar./Apr. 2008.

ARSHAD AHMAD received the Ph.D. degree in computer science and technology from the Beijing Institute of Technology, China, in 2018. He is currently an Assistant Professor of computer science with the Department of Computer Science, University of Swabi, Ambar, Pakistan. His research interests include requirements engineering, text mining, sentiment analysis, and machine learning.

CHONG FENG received the Ph.D. degree in computer science from the University of Science and Technology of China, Hefei, in 2005. He is currently an Associate Professor of computer science and technology with the Beijing Institute of Technology, Beijing. His current research interests focus on social media processing, information extraction, and machine translation.

KAN LI received the Ph.D. degree in computer science from the Beijing Institute of Technology, China, in 2003, where he is currently a Professor of computer science and technology. He has published over 40 technical papers in peer-reviewed journals and conference proceedings. His research interests include machine learning, data mining, and distributed systems.

SYED MOHAMMAD ASIM received the Ph.D. degree in applied statistics from the University of Gottingen, Germany, in 2008. He is currently the Chairman and an Associate Professor of Statistics with the University of Peshawar, Peshawar. His current research interests focus on biostatistics, econometrics, multivariate analysis, and social statistics.



TINGTING SUN received the bachelor's degree in computer science from the University of Shenyang, Shenyang, in 2014. She is currently pursuing the master's degree with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing. Her research interests focus on natural language processing, information extraction, and information retrieval.