# An Empirical Study on How iOS Developers Report Quality Aspects on Stack Overflow

Arshad Ahmad, Kan Li, Chong Feng , and Tingting Sun

***Abstract*—Software developers around the globe are actively asking a question(s) and sharing solutions to the problems related to software development on Stack Overflow - a social question and answer (Q&A) website. The knowledge shared by software developers on Stack Overflow contains useful information related to software development such as feature requests (functional/non-functional), code snippets, reporting bugs or sentiments. How to extract the functional and non-functional requirements shared by mobile application developers on social/programming Q&A website Stack Overflow has become a challenge and a less researched area. To understand the problems, needs, and trend in the iOS mobile application development, we evaluated the quality requirements or non-functional requirements (NFRs) on Stack Overflow posts. To this end, we applied Latent Dirichlet Allocation (LDA) topic models, to identify the main topics in iOS posts on Stack Overflow. Besides, we labeled the extracted topics with quality requirements or NFRs by using the wordlists to evaluate the trend, evolution, hot and unresolved NFRS in all iOS discussions. 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.**

***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 addressed properly in every software development life cycle. Requirements need to be properly elicited, stated, verified & validated, and maintained as needed [1]-[4]. In the 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. These social media and online platforms include Stack Overflow Q&A community [5], Twitter, issue tracking systems, and mobile application stores like Google, and Apple [6].

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 [7]. 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. It is highly probable that the content shared on Stack Overflow posts contain statements about mobile application development, tools, and product qualities [6].

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 [8], and inadequately addressing them will mostly result in project failure and increase in rework cost [1], [4]. Thus, NFRs should be addressed in early stages of any software development to avoid any underlying problems. However, eliciting entirely complete and precise set of NFRs or quality requirements is challenging [9], [10].

The recent years have witnessed an 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 [11], [12]. Some of the research studies considered the issues faced by mobile application developers (e.g., [11], [13]-[15]), and others have focused on general software developers issues (e.g. [5], [16]-[19]). 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.

Thus, 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 mobile application development. We specifically restrict the scope 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 on Stack Overflow?

**RQ2:** What are the most important non-functional requirements discussed in unanswered iOS Stack Overflow posts?

Manuscript received July 17, 2018; revised September 19, 2018. This work was supported by the National 863 Project, China, under Research Grant 2015AA015404.

The authors are with School of Computer Science & Technology, Beijing Institute of Technology, Beijing, 100081, China (e-mail: fengchong@bit.edu.cn, 2220170592@bit.edu.cn).

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

The research questions aim to identify the important NFRs or quality requirements discussed on Stack Overflow related to iOS mobile application development. We use huge scale of iOS posts data available on Stack Overflow to investigate not only the most the important NFRs along with their trend but also the common problems faced by iOS developers. Since Stack Overflow is daily used by thousands of experienced developers and their discussions trends mostly represents the current needs of users and market trends. This will ultimately help 1) iOS developers to know what are the most important NFRs and issues that needs 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, and 4) assist software engineering academics and industry in identifying the problematic areas for iOS developers that needs further research and attention.

The remainder of this paper is organized as follows: Section II describes the data and research approach used for this study. The results and discussion of the study are explained in Section III. Finally, Section IV provides the conclusion and point out avenues for future research.

## II. Data and Approach

In this section, we describe how we carried out our study in three steps. At firstly, we extracted the iOS posts data from Stack Overflow, and then applied some preprocessing steps on the extracted data. Then, we applied topic model Latent Dirichlet Allocation [20] 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 [21], especially suitable for the domain of software engineering.

*Step 1: Extract and select SO posts:* To address the research questions of our study, we extracted the posts and comments provided by a programming/social Q&A website Stack Overflow[2] from 31st July 2008 up to 31st August 2017. The Q&A on Stack Overflow 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. We used the Python library Beautiful Soup[3] to extract only those posts tagged "iOS," totaling about 525K posts and 985K comments. To address the RQ1, we mainly used two types of corpus: the "title" & "body" of iOS posts combined with the "text" of the comments and the other type only have the "title" and "body" of the iOS posts. We compare the outcomes of the two types of corpus. For addressing RQ2, we only extract the "title" and "body" of the unanswered questions from iOS posts totaling approximately 228K. For addressing RQ3, we utilize both the "title" and "body" of iOS posts and the "title" and "body" of the unanswered questions. Fig. 1 depicts 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.

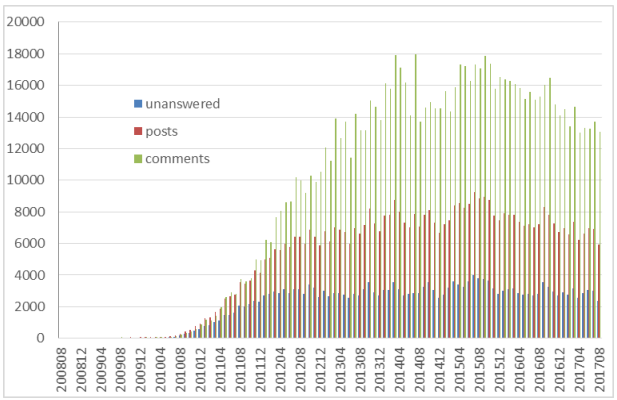


Fig. 1. Overall dataset used.

After successful desired data extraction, 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. Afterwards, to further refine the information in the data we performed tokenization, stop words removal, and case unification respectively.

*Step 2: LDA Topic Modeling:* In this research study we use and construct the topic model LDA by sklearn [22]. The topic model LDA is applied to extract the topics of our corpus. In LDA topic model, the topic represents the conditional probability distribution of words in a particular vocabulary. To apply 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 parameter. For our experimental work, we selected the number of topics parameter $K$=20 for each of the specified periods 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 [21], [23]. Besides, we did not change the default settings for $N$=1000, $\alpha$= 0.05, and $\beta$=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.

*Step 3: Perform 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 word list used in this study is the exp2; especially suitable for the domain of software engineering [21]. We match the words of the extracted

[2] https://archive.org/details/stackexchange
[3] https://www.crummy.com/software/BeautifulSoup/

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 NFRs.

*Step 4: Validating the Corpus:* To assess our 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 participants were asked to label one year data (January 2016-December 2016), and they finished the labeling task in about one week. 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 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 software engineering domain.

## III. Results and Discussion

### A. 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 run our approach on it. Then, we compare the outcome with the results generated through manual validation set. The calculation criteria for recall and precision rate are given in Equation 1 and 2 respectively.

$$\text{RecallRate} = \text{Ndetected} / \text{Ntotal} \quad (1)$$

$$\text{PrecisionRate} = \text{Ndetected} / \text{Ndetectedall} \quad (2)$$

where *Ndetected* represents the number of precise NFRs labels (i.e., the NFRs or quality requirements label corresponds the manual annotation), *Ntotal* represents the whole number of the manual NFRs labels in our testing set, *Ndetectedall* 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 *Ndetected* is 2 (usability and reliability), the value of *Ntotal* is 3 (usability, reliability, and functionality), and finally, the value of *Ndetectedall* 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.

Fig. 2 depicts the calculated recall rate and the precision rate for each period (month) of our results. It is evident in Fig. 2 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.

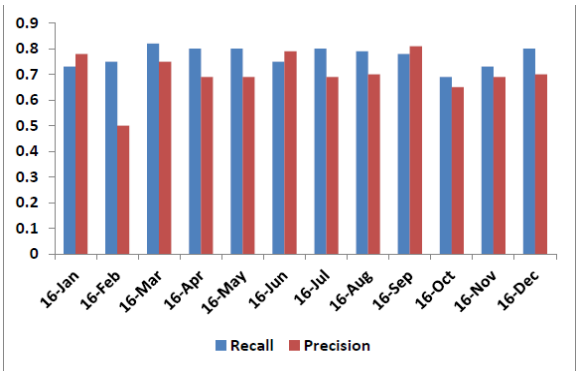



Fig. 2. Calculated percentage of recall and precision rate of NFRs labeling.

### B. Results of RQ1

Fig. 3 depicts the rate of six different quality requirements or NFRs using the posts data of iOS. The x-axis represents the rate or value of the six corresponding quality requirements or NFRs (i.e., the extracted topics labeled by the six corresponding quality requirements or NFRs divided by the total number of extracted topics). The y-axis represents the six respective quality requirements or NFRs. It is evident in Fig. 3 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 the 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 application development. 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).

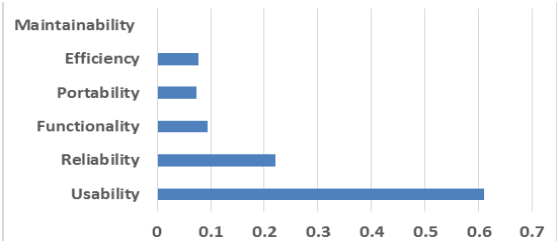



Fig. 3. Rate of distribution of different NFRs in posts only.

### C. Results of RQ2

To address the RQ2, we primarily focused on all

unanswered iOS questions on Stack Overflow with the aim to investigate the unsolved critical problematic domains. It in return will be more helpful for the iOS application developers to highlight the most challenging issues they face during development. Fig. 4 depicts the distribution rate of six quality requirements or NFRs about all iOS unanswered or unaddressed questions. It is evident in Fig. 4 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 means 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 development. The least frequent are efficiency and maintainability problems in iOS development developers face, or they can better handle it easily those issues. In future, more research is needed in this area to investigate in detail the nature of all those critical issues so that the academic and industry should come up with possible solutions.

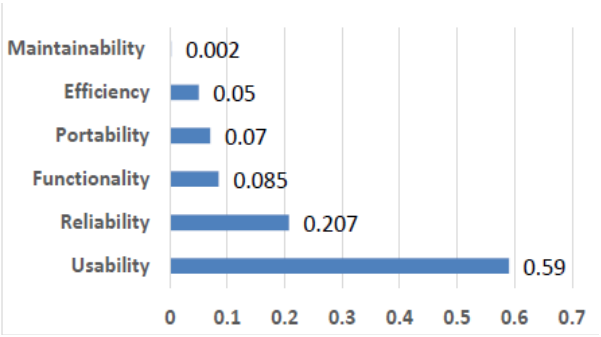


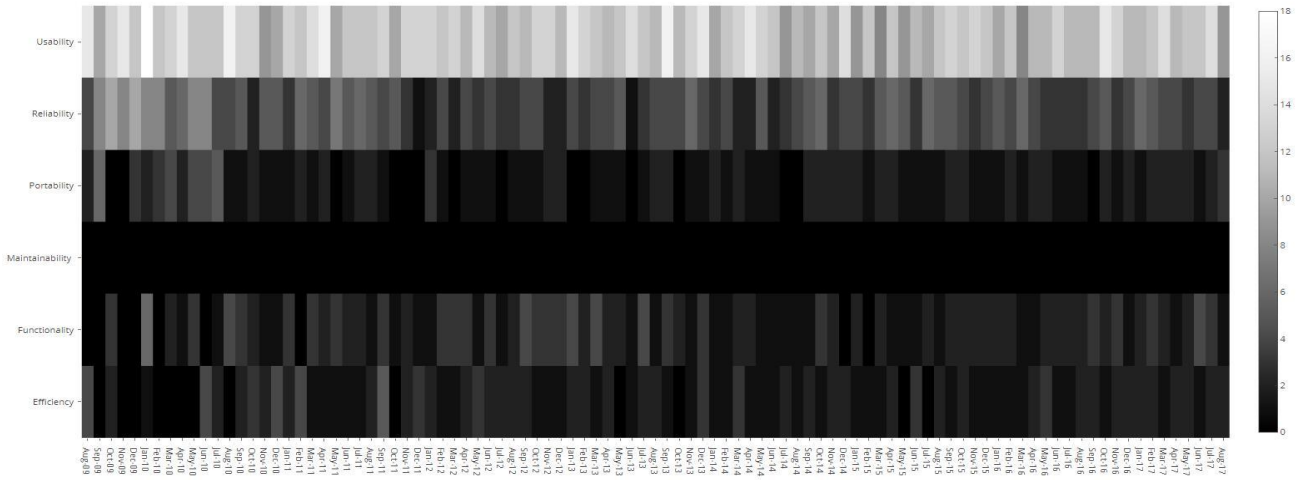

Fig. 4. Rate of distribution of different NFRs in unanswered posts or questions.

### D. Results of RQ3

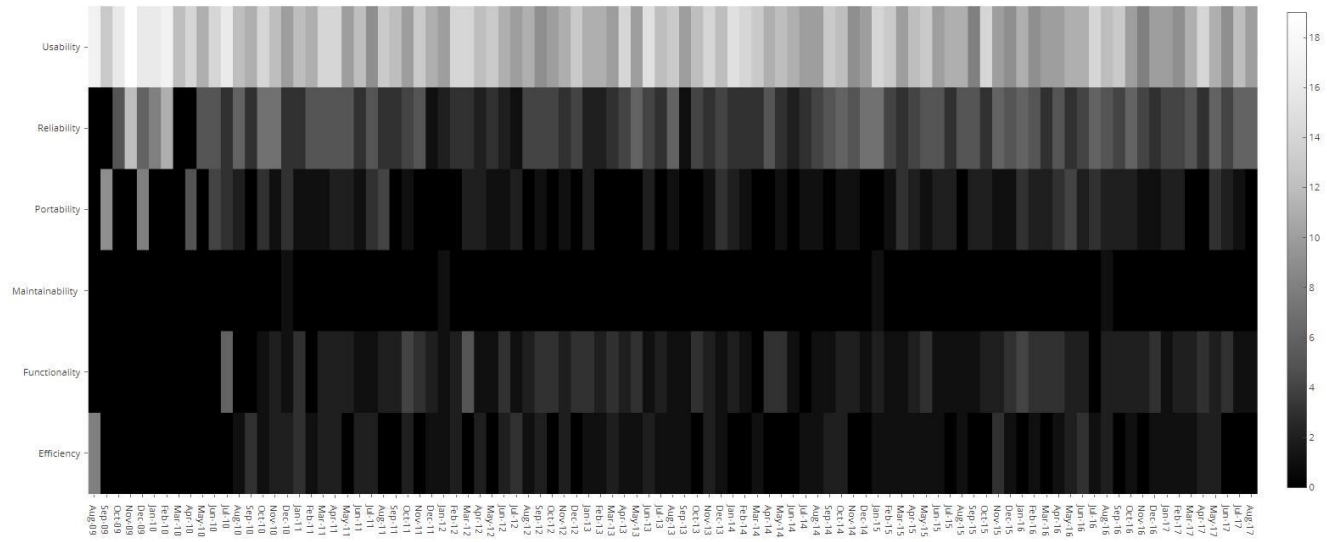
To address RQ3, we only use posts data because the outcome of RQ1 determined that using posts alone and using posts along with the comments have same results. Through our approach, we label the extracted topics of iOS posts, and revealed that most of the topics are labeled with one NFR is approximately 62.39%, more than one NFR are approximately 17.93%, and approximately 19.68% are labeled with "none."

Fig. 5(a) & Fig. 5(b) depicts the gray-scale image of the six different quality requirements or NFRs frequencies over the period. The cell corresponds to a 30-day period. The higher intensity or deep color of a grid cell represents 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 the passage of time. Fig. 5(a) & (b) not only depicts visually the evolution of each of the six different NFRs with the passage of time but also depicts the trend of hot or not hot NFRs in a particular timeline. Fig. 5(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 the time. It is also evident that efficiency, functionality, and portability frequency trends are up and down over the time. The frequency of maintainability is least among all and stays constant from the start until the end of the period.

Fig. 5(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 the passage of 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 Fig. 5(a) and Fig. 5(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.




(a) iOS Posts




(b) Unanswered iOS Posts

Fig. 5. Rate of frequencies of NFRs over time.

## IV. Conclusion and Future Work

We used LDA topic model to identify and evaluate the NFRs discussed iOS development on Stack Overflow posts. 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 comments yielded similar results. The most problematic areas left unresolved lies in usability, reliability, and functionality, which hints of more work in future these areas to improve iOS development. 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. 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 future, we welcome other researchers and would like to focus deeply on specific iOS development tool 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. International 2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*, Izmir, 2012, pp. 297-306.

[2] H. Khan, A. Ahmad, C. Johansson, and M. A. Alnuem, "Requirements understanding in global software engineering industrial surveys," in *Proc. 2011 International Conf. on Computer and Software 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," *Research Journal of Applied Sciences, Engineering and Technology*, 2012.

[4] A. Ahmad and H. Khan, "The importance of knowledge management practices in overcoming the global software engineering challenges in requirements understanding," Master Thesis Research, Blekinge Institute of Technology, Sweden, 2008.

[5] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?: Nier track," in *Proc. 2011 33rd International Conference on Software Engineering (ICSE)*, 2011, pp. 804-807.

[6] E. C. Groen, S. Kopczy ńska, 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 International Requirements Engineering Conference*, 2017, pp. 80-89.

[7] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Proc. IEEE International Conference on Requirements Engineering*, 2014, pp. 153–162.

[8] D. Ameller, C. Ayala, J. Cabot, and X. Franch, "Non-functional requirements in architectural decision making," *IEEE Software*, vol. 30, pp. 61–67, 2013.

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

[10] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, "Nonfunctional requirements in industry-three case studies adopting an experience-based nfr method," in *Proc. IEEE International Conference on Requirements Engineering*, 2005, pp. 373–382.

[11] C. Rosen and E. Shihab, "What are mobile developers asking about? A large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, pp. 1192-1223, 2016.

[12] A. Ahmad, C. Feng, M. Tao, A. Yousif, and S. Ge, "Challenges of mobile applications development: Initial results," in *Proc. 8th IEEE International Conference on Software Engineering and Service Science*, Beijing, China, 2017, pp. 464-469.

[13] M. Linares-V ásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development issues using stack overflow," in *Proc. the 10th Working Conference on Mining Software Repositories*, 2013, pp. 93-96.

[14] M. Rebouc ąs, G. Pinto, A. Serebrenik, and F. Castor, "An empirical study on the usage of the swift programming language," in *Proc. IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering*, 2016, pp. 634-638.

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

[16] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, pp. 619-654, 2014.

[17] 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. 2015 IEEE*/ACM *12th Working Conference on Mining Software Repositories*, 2015, pp. 446-449.

[18] G. Pinto, W. Torres, and F. Castor, "A study on the most popular questions about concurrent programming," in *Proc. the 6th Workshop on Evaluation and Usability of Programming Languages and Tools*, 2015, pp. 39-46.

[19] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *Proc. the 11th Working Conference on Mining Software Repositories*, 2014, pp. 112-121.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993-1022, 2003.

[21] 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 Working Conference on Mining Software Repositories*, 2011.
[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.
[23] A. Hindle, M. W. Godfrey, and R. C. Holt, "What's hot and what's not: Windowed developer topic analysis," in *Proc. IEEE International Conference on Software Maintenance*, 2009, pp. 339–348.

**Arshad Ahmad** received his MS degree in software engineering from Blekinge Institute of Technology, Sweden in 2008. He is currently a PhD student at School of Computer Science and Technology, Beijing Institute of Technology, China. His research interests include requirements engineering, text mining, sentiment analysis and machine learning.

**Kan Li** received his PhD degree in computer science from Beijing Institute of Technology, China in 2003. He is currently a professor of computer science and technology in Beijing Institute of Technology, Beijing. 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.

**Chong Feng** received his PhD degree in computer science from the University of Science and Technology of China, Hefei, in 2005. Now he is an associate professor of computer science and technology in Beijing Institute of Technology, Beijing. His current research interests focus on social media processing, information extraction, and machine translation.

**Tingting Sun** received her bachelor's degree in computer science from the University of Shenyang, Shenyang in 2014. Currently, she is a master student at 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.