Which Non-functional Requirements do Developers Focus on?

An Empirical Study on Stack Overflow using Topic Analysis

Jie Zou[b], Ling Xu[a,b,*], Weikang Guo[b], Meng Yan[b], Dan Yang[b], Xiaohong Zhang[a,b]
a. Key Laboratory of Dependable Service Computing in Cyber Physical Society Ministry of Education, Chongqing 400044, PR China
b. The School of Software Engineering, Chongqing University, Chongqing (401331), PR China.
xuling@cqu.edu.cn

***Abstract*—Programming question and answer (Q&A) websites, such as Stack Overflow, gathered knowledge and expertise of developers from all over the world, this knowledge reflects some insight into the development activities. To comprehend the actual thoughts and needs of the developers, we analyzed the non-functional requirements (NFRs) on Stack Overflow. In this paper, we acquired the textual content of Stack Overflow discussions, utilized the topic model, latent Dirichlet allocation (LDA), to discover the main topics of Stack Overflow discussions, and we used the wordlists to find the relationship between the discussions and NFRs. We focus on the hot and unresolved NFRs, the evolutions and trends of the NFRs in their discussions. We found that the most frequent topics the developers discuss are about usability and reliability while they concern few about maintainability and efficiency. The most unresolved problems also occurred in usability and reliability. Moreover, from the visualization of the NFR evolutions over time, we can find the trend for each NFR.**

***Index Terms*—Non-functional requirements (NFRs), Topic model, Latent Dirichlet allocation (LDA), Stack Overflow.**

I. INTRODUCTION

With the development of the scale and size of software, people are more concerned about software quality requirements. Non-functional requirements (NFRs) describe important constraints upon the development and behavior of software, they should be considered as early as possible, otherwise they may cause some latent problems of software artifacts like unstable and low quality. It may become very complex and expensive to address them at later stages of software development, especially for large systems.

Previous works on topic analysis show that extracting the topics was useful in software maintenance, and it's helpful for understanding the software development activities[1][2]. The topics extracted from the corpus summarized the key concepts in the corpus, such as source code, text descriptions, and commit comments. The topic model, Latent Dirichlet Allocation (LDA)[3], can find the topics of the corpus, it created a multinomial distributions of words to represent the topics.

Since the extracted topics are abstract and hard to understand, we need to devise appropriate labels for these topics to make it easy to use in discussion and analysis. We annotate them with the NFRs because the topic trends often corresponded to NFRs[4]. Through that we discover the focus and needs of the developers, propose the visualizations of NFRs what developers discussed during the development activities. Managers may also be able to grasp the topic-related activities of the developers more easily.

In this paper, we extract the topics of the corpus using the topic model LDA, and then annotate the topics with the NFRs labels by a special wordlist. After that we analyze the focuses, unresolved problems, evolutions and trends of the NFRs. We divide the corpus into time-windows when extracting the topics of the corpus. We partition the whole corpus into 30 day periods because the period size of 30 days is smaller than the time between minor releases but large enough for there to analyze[1]. We divide the corpus into time-windows rather than regard it as a whole because many topics are quite local. This also allows us to explore the evolution of the NFRs over time. When annotating the NFRs, we utilize the standards of ISO9126, which describes six high-level NFRs: maintainability, functionality, portability, efficiency, usability, and reliability.

We use the data posts from Stack Overflow, and in our analysis, we also distinguished between posts with or without comments. Our study aim to answer the following three research questions:

RQ1) what's the hot and not hot NFRs? In other words, which NFRs are discussed most and least frequently?

RQ2) which NFRs questions remain unanswered at most? Which NFRs are focused most in unresolved problems?

RQ3) what are the evolutions and trends of the NFRs with respect to time?

By answering these questions, we want to provide much immediately useful detail about the software development activities. Since Stack Overflow is worldwide and very popular with tens of thousands of developers, the developer discussions trends are symbiotic with the market trends. We hope to

978-0-7695-5594-2/15 $31.00 © 2015 IEEE
DOI 10.1109/MSR.2015.60

indicate the focus and the needs of developers, give a guide to developers which NFRs they should focus on, such as usability and reliability, others should pay attention to which fields in software development activities if they want to offer help to developers. We expect that the evolution of the NFRs will provide the changes in developer focus as they change over time. The trend lines give us an indication of the rise or fall of developer interest. The analysis also provides managers, maintainers or commercial vendors an enhanced understanding of the history of software development. It allows managers and maintainers to track the key aspects of development activities, commercial vendors to assess the adoption rate of their products, and to understand usage trends. Moreover, the research provides software engineering researchers with immediate knowledge of some of the possible troublesome areas. Our technique labeled the topics with NFRs is also extendable to help with automatic tagging or summarization.

II. Data and Approach

In this section we describe how to annotate the NFRs. Our approach consists of three key steps. Firstly, we extract the data from Stack Overflow, and then preprocess the extracted data. Secondly, we construct a topic model LDA to extract the topics of the corpus. Finally, we label the topics with the NFRs by our wordlists.

A. Data Extraction Process

To address the above research questions, we used the posts and comments of Q&A site Stack Overflow from July 31, 2008 to September 14, 2014 provided by the MSR challenge[5], Stack Overflow is a global site which features questions and answers on a wide range of topics in computer programming. We used the java SAX parser to extract the data of posts and comments, totaling about 21.7 million posts and 32.5 million comments. For our analysis, to the first research question, we utilize two types of corpus. One is the "title" and "body" of posts combined with the "text" of comments. The second one only contains the "title" and "body" of posts. We compare the results between the two corpuses. For the second research question, we extract the "title" and "body" of the unanswered questions from the posts, this totals about 921K. The third research question we use the "title" and "body" of posts and the "title" and "body" of the unanswered questions. Figure 1 shows the detailed data of each month (period), with the month as the x-axis, and the number of posts or comments as the y-axis, the highest reaches 0.95 million.

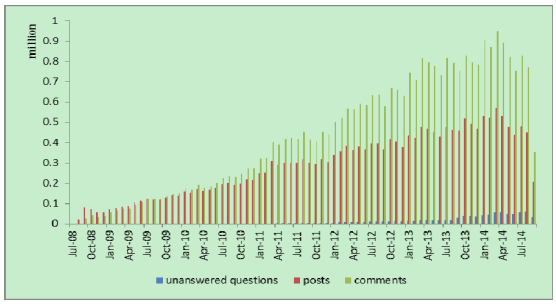


Fig. 1. The dataset.

After data extraction, we preprocess the extracted data as follows. At first, we remove the periods whose number of posts less than 100, since too few posts are useless for the analysis. And then a few of remaining preprocess steps are carried out to further refine the information, including word segmentation, stop words removal, and case unification.

B. Topic Modelling

We apply LDA to extract the topics of the corpus. Described in mathematical language, the topic is the conditional probability distribution of words in the vocabulary. LDA requires the number of topics parameter K, the number of iteration N, the dirichlet parameter α and β as inputs. In our experiment, we choose the number of topics as 20 for each period because duplicate words from different topics are infrequent when K=20, maybe 20 is not necessarily the optimal choice but it is an appropriate value for NFRs analysis[1][4]. And we use the default settings N=1000, α=2.5, β=0.01. The output of LDA in our experiment is a matrix M where rows correspond to the K topics of posts or comments and columns correspond to the words of topics. In this paper, we use MALLET[6] to construct the topic model LDA.

C. NFRs Labeling

In order to analyze the NFRs, we need to annotate these topics with NFRs labels. We use the ISO quality model, ISO9126 as the taxonomy of NFRs. Although we have no evidence to state that ISO9126 is a correct and detailed standard, it constitutes the most widely used software quality model at present. Therefore, we consider it is sufficient for the purposes of this research. We associate each NFR with a list of keywords, called wordlists. We contrast the words of the topics with the words of our wordlists. If any of the words of the topics match any of the wordlist's words, we labelled the topic with the corresponding NFRs. If there are no matched words, we labeled the topic with "none", we think this topic is not associated with any of the NFRs. Each topic can be labeled with one or more NFRs. The wordlists[1] we use, which is specific for software field, is the exp2 from [4].

D. Creating a Validation Corpus

To evaluate the annotated results, we invited four PhD candidates who researched in software engineering to label the topics manually as a validation set. The annotators spent about five work days to label the data from Aug 2013 to Aug 2014, totaling 12 months. They look at the topics of each period and the words of each topic, using their expertise and then suggested the appropriate label to the topics, the four annotators did not annotate each other's annotations, and the labels they used are also one or more of the six NFRs of ISO9126. They can also label the topics with "none" if they think there are no NFRs associated with the topics. They also use the original files related to the topics being annotated as auxiliary information when labeling the topics. For example, a topic of Sept. 2014 consisting of the word "code exit view error

[1] http://softwareprocess.es/y/neil-ernst-abram-hindle-whats-in-a-name-wordlists.tar.gz

null method init custom button click ui fault stable input", they tagged the topic with usability and reliability. We suppose the manual labeled topics are correct.

III. Analysis of The Results

A. Accuracy of Evaluation

To the problem of how well our approach work, we use the recall and precision rates as standard metrics to evaluate the accuracy of the NFRs labeling. We run our approach on the testing set, which is the post data from Sept 2013 to Aug 2014. Next, we compare the results with the manual validation set.

$$RecallRate = N_{detected} / N_{total.} \quad (1)$$

$$PrecisionRate = N_{detected} / N_{detectedall.} \quad (2)$$

Where $N_{detected}$ is the number of correct NFRs labels (the NFRs label matches the manual annotation), N_{total} is the total number of the manual NFRs labels in the testing set, $N_{detectedall}$ is the total number of NFRs labels in the experimental results using our approach (including the correct and incorrect). For example, if we label the topic with usability, efficiency and functionality, and the manual validation set labels the topic with usability, efficiency and reliability, $N_{detected}$ is 2 (usability and efficiency), N_{total} is 3 (usability, efficiency, and reliability), $N_{detectedall}$ is 3 (usability, efficiency, and functionality), so that the recall rate is 2/3≈66.7%, the precision rate is 2/3≈66.7%.

Figure 2 shows the recall rate and the precision rate of our results. To each period, we calculate a recall rate and a precision rate. From figure 2, we see that the highest recall rate and precision rate of our results reaches 80%, averaging 75.9% and 68.7% respectively.

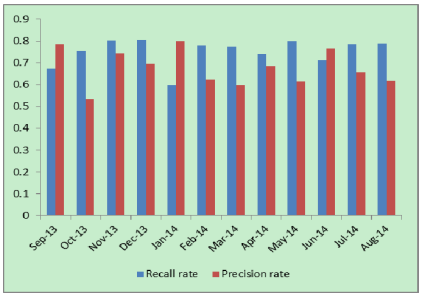


Fig. 2. The recall rate and the precision rate of our NFRs labeling.

B. Results RQ1

Figure 3 shows the rate of different NFRs using the data posts, with the x-axis is the rate of the corresponding NFRs (the topics labeled by the corresponding NFRs divide by the total topics), the y-axis is the corresponding NFRs. From figure 3, we see that the labels with the most topics are usability and reliability, and then functionality and portability. We did not see many results for efficiency or maintainability. That is, the NFRs the developers discussed most frequently are usability and reliability, and the least they discussed are efficiency and maintainability. When they are coding, they mainly focus on usability, and then reliability and functionality. In contrary, they almost pay no attention to efficiency or maintainability.

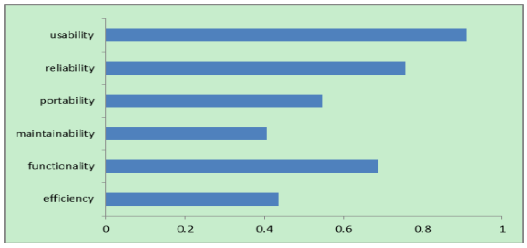


Fig. 3. The rate of different NFRs using posts only.

And we also compare the results with the posts combined with the comments. Figure 4 shows the results using posts and comments data. From figure 4, we see that the relative distribution of the NFRs is almost the same as the results of using the posts only. Usability and reliability are the highest, and then functionality and portability. The lowest are maintainability and efficiency.

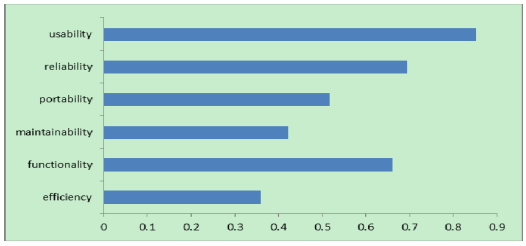


Fig. 4. The rate of different NFRs using posts and comments.

C. Results RQ2

For the research question 2, we analyzed the unanswered questions from Stack Overflow so as to explore the unresolved problem domains. Through this, we can help more to the developers. Figure 5 shows the rate of different NFRs about unanswered questions. From this, we see the most topics remain unanswered are labeled with usability and reliability, and then functionality and efficiency. The fewest are portability and maintainability. The problems remain unresolved mostly are in the usability and reliability domains. As to portability and maintainability, they usually can work this out by themselves, or we can say they rarely meet the problems of portability and maintainability. That is to say there are possible troublesome areas in usability and reliability, to help the developers more, we should concern ourselves more with the issues of usability and reliability.

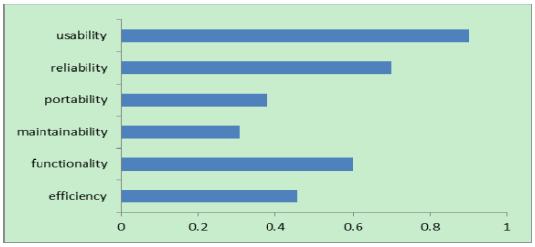
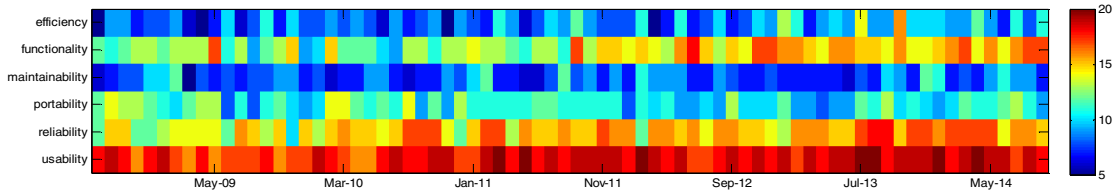


Fig. 5. The rate of different NFRs about unanswered question.

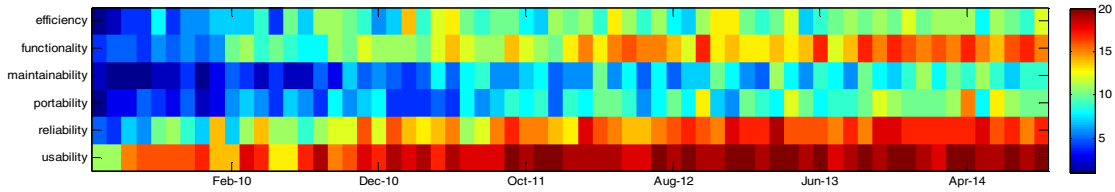
D. Results RQ3

Since the RQ1 concludes that the results of using posts only and using posts combination with the comments is almost the same, in this research question, we use posts data only to cut down the data size. We label the topics of posts using our approach, and we find that most of the topics are labeled with more than one NFR, about 92.5%. The topics labeled with only one NFR is about 4.9%, and about 2.6% labeled with "none".

Figure 6 shows the gray-scale image of the time-lines of NFRs frequencies. Each cell represents a 30-day period. Grid cell intensity (saturation) is mapped to label frequency relative to the largest label count of all NFRs. Deeper color indicates the greater frequencies. From this figure, we can see the visualization of evolution of each NFR with time, and we also can see the hot or not hot NFRs in a certain period. Figure 6(a) shows the results of the posts, and we see that all six NFRs change over time, but the trends are not the same. We observe that the efficiency, functionality and the reliability rise slowly while the maintainability is up and down as time goes on. At the same time the portability drops off. And the usability is almost stable at a high frequency. Figure 6(b) shows the results of the unanswered questions. Four NFRs, functionality, maintainability, portability, and reliability increase with time, the efficiency fluctuates and usability remains almost stable. From figure 6(a) and figure 6(b), we see that functionality and reliability are increasing not only for posts but also for unanswered questions. Moreover, both of the usability topics are almost stable at a high frequency. These suggest that functionality and reliability will increase the attention of the developers and the usability will remain hot in the next few years. There are also some peaks in figure 6, this shows that the maintenance activity is not necessarily strictly increasing or decreasing with time.



(a) Posts



(b) Unanswered questions

Fig. 6. The frequencies of different NFRs over time.

IV. Conclusion

We use a topic model to analyze the NFRs of Stack Overflow data. We found that the developers focus most on usability and reliability, and which they are less concerned about maintainability and efficiency. We also compared the results of using posts only with the results of using posts and comments, and found that the results of the two settings are almost the same. The most problems remain unresolved are also related to usability and reliability; they need more help in the usability and reliability areas. We also analyze the trends of the different NFRs on the posts and the unanswered questions. We found that the NFRs are changing over time; functionality and reliability increase and the usability always remain hot.

Acknowledgment

The work described in this paper was partially supported by the National Natural Science Foundation of China (Grant no. 91118005, 61173131), Changjiang Scholars and Innovative Research Team in University (Grant Nos. IRT1196), and the Fundamental Research Funds for the Central Universities (Grant Nos. 106112013CDJZR090005).

References

[1] A. Hindle, M. W. Godfrey, and R. C. Holt, "What's hot and what's not: Windowed developer topic analysis," in Software Maintenance, 2009. ICSM 2009. IEEE International Conference on, 2009, pp. 339–348.

[2] 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," submission http//softwareprocess. es/whats-in-a-name, vol. 125, pp. 150–155, 2010.

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

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

[5] A. T. T. Ying, "Mining Challenge 2015: Comparing and combining different information sources on the Stack Overflow data set," in The 12th Working Conference on Mining Software Repositories, 2015, p. to appear.

[6] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002.