DATA-DRIVEN MODEL FOR NON-FUNCTIONAL REQUIREMENTS IN MOBILE APPLICATION DEVELOPMENT

Salisu Garba[1] Babangida Isyaku[2] and Mujahid Abdullahi[3]

[1,2,3]Department of Mathematics & Computer Science, Sule Lamido University, Kafin Hausa. Jigawa State.

ABSTRACT

The incredible development in the utilization of smartphones has driven the development of billions of software applications famously known as 'apps' to accomplish roles outside phone call and SMS messages in the day-to-day lives of users. Current assessments show that there are a huge number of applications developed at a meteor pace to give clients a rich and quick client experience. Mobile apps users are more concerned about stability and quality now more than ever despite the increase in the scale and size of apps. As such, mobile apps have to be designed, built, and produced for less money (maintainability, portability, and reusability), with greater performance, reliable security and fewer resources (efficiency) than ever before. This paper aimed at providing support for mobile application developers in dealing with the ever-eluding non-functional requirements by proposing a data-driven model that simplifies the non-functional requirements (NFR) p in the development of an application for mobile devices. The study tries to find out if NFR can be treated the same way as functional requirements in mobile application development. Finally, this paper shows the experimental evaluation of the proposed data-driven model of dealing for non-functional requirements in the development of mobile apps and the results obtained from the application of the model are also discussed.

KEYWORDS

Non-Functional Requirements, Mobile Application Development, Data-Driven Requirement Engineering, Requirement Modelling

1. INTRODUCTION

The modern incarnation of the mobile applications began in 2007 when a 1st generation of iPhones together with a concentrated market for applications called the 'Application Store', through which the end-users can download and install different applications. Not long after in 2008, Google set-up another platform called Google Play, officially known as (Android) to rival the 'Application Store' and boost-up the 'Android Market'. Microsoft and BlackBerry also tag along with comparable application markets for mobile apps. With these application markets, now the mobile application engineers have a considerably bigger client base to pitch to. It is assessed that there are as of now 2.6 Billion smartphone users [12].

However, seeing that mobile applications are developing into a more intricate, shifting away from inexpensive frivolous applications to more business-critical uses, it will be vital to apply software engineering processes to guarantee the development of stable, safe and qualitative mobile applications [21]. Having said that, numerous "classic" software engineering techniques can easily be transformed into the mobile application domain, requirement engineering is among the areas that need further exploration.

One of the well-established differences between functional and non-functional requirements is the functions of the system and how the system shall carry-out the functions. This difference has a huge influence on the elicitation, documentation, analysis, and validation of both sets of requirements practically as demonstrated by several types of research such as [3]. There is a scarcity of generally accepted approach for elicitation, documentation, analysis, and validation of non-functional requirements practically, as a result, non-requirement is more often than not describe imprecisely, usually not quantified, thereby generating a result that is complicated even before its tested.

Moreover, non-requirements are frequently retrofitted in the mobile application development process otherwise pursue in concurrently with, but independently from, functional requirements and, therefore, are completely managed with diminutive analysis. Mobile application developers, requirements analysts, and managers could systematically use explicit and implicit user feedback in an aggregated form to support non-functional requirements decisions.

This paper proposed a data-driven model for dealing with non-functional requirements in mobile application development so as to enhanced quality as well as effectiveness in carrying out requirements' engineering activities including requirements priority and traceability for mobile application development. The remaining of this paper shows the background and related works in section 2. The proposed model is illustrated and discussed in section 3. The evaluation and results obtained from the application of the model are discussed in section 4. The conclusion and future work are discussed in section 6.

2. BACKGROUND & RELATED WORK

The ever-increasing complexity of devices, the escalating market for applications and the growing edge of wireless networks all work together making mobile application development industry with great potential. Consequently, powerful development tools and frameworks are fashioned to greatly simplify the task of implementing a mobile application. However, they are predominantly focused on the individual developer who is trying to create an application as quickly as possible. In an effort to gain a better understanding of development practices for mobile applications, recent surveys such as that conducted by [6] have shown that mobile application developers adhered relatively well to recommendations or the most effective method but rarely used any formal development processes, moreover, there is a lack of persistence effort by developers in tracking developed apps so as to gather new metrics for future use.

Several recent studies investigating non-functional requirements in mobile application development, [12] discover that BlackBerry applications are bigger and depend more on foreign libraries, while, Android applications depend intensely on the Android stage. [10] proposed a paradigm to deal with biases of requirement specification tools as the majority of requirements specification tools are more suitable for functional requirements than for non-functional requirements. Reliability, availability, maintenance, and performance (RAMP) requirements are left unstipulated, or at best vaguely stipulated, which makes requirements specifications more of an art than a science. Furthermore, the cost of testing for RAMP requirements is frequently excessive.

According to [12], [4], there have been a lot of focuses on a wide range of information that can be mined from the application markets, with the application themselves being only one sort of information. For instance, [8] utilized etymological principles to identify features demands from user application reviews to generate more abstract requirements. [1] mine opinions, ideas and facts from user application reviews with a specific end goal to revise requirements, the results of the comparison between automatically extracted requirements and manually extracted requirements are coherent in requirements specifications. [5] utilize natural language processing (NLP) methods to discover features in application review and use sentiment analysis to establish how users experience about application features.

Researches on mobile application requirements have been mostly restricted to off-the-shelf natural language processing (NLP) tools that not design to mine text from user-reviews (which can be extremely concise, have a tendency to be vastly unstructured, and have grammatical mistakes). However, [2] proposes AR-Miner; a ranking algorithm that helps in prioritizing the identified user-reviews which is consistent to real developers, and recognizing traceability interfaces between user-reviews and application features.

The work nearest to our own is the work by [21], which looks at the non-functional requirements through the mobile application Developers' eyes. The fundamental contrast between our work and their work is the way that we find issues through mining inquiries from tack overflow to supplement the data extracted from app reviews posted by users in different app stores. Additionally, our work supplements theirs by including more profundity into their investigation, i.e., our approach takes a look at the issues by analyzing genuine inquiries asked by users, instead of getting information from the developers.

3. THE DATA-DRIVEN MODEL FOR NFR IN MAD

After going through a considerable amount of literature on requirement engineering, mobile application development, non-functional requirements in the development of mobile applications, challenges in non-functional requirements identification and elicitation, the proposesed data-driven model of dealing with non-functional requirements in mobile application development is shown in Figure 1 below. The activities in each layer are discussed below.

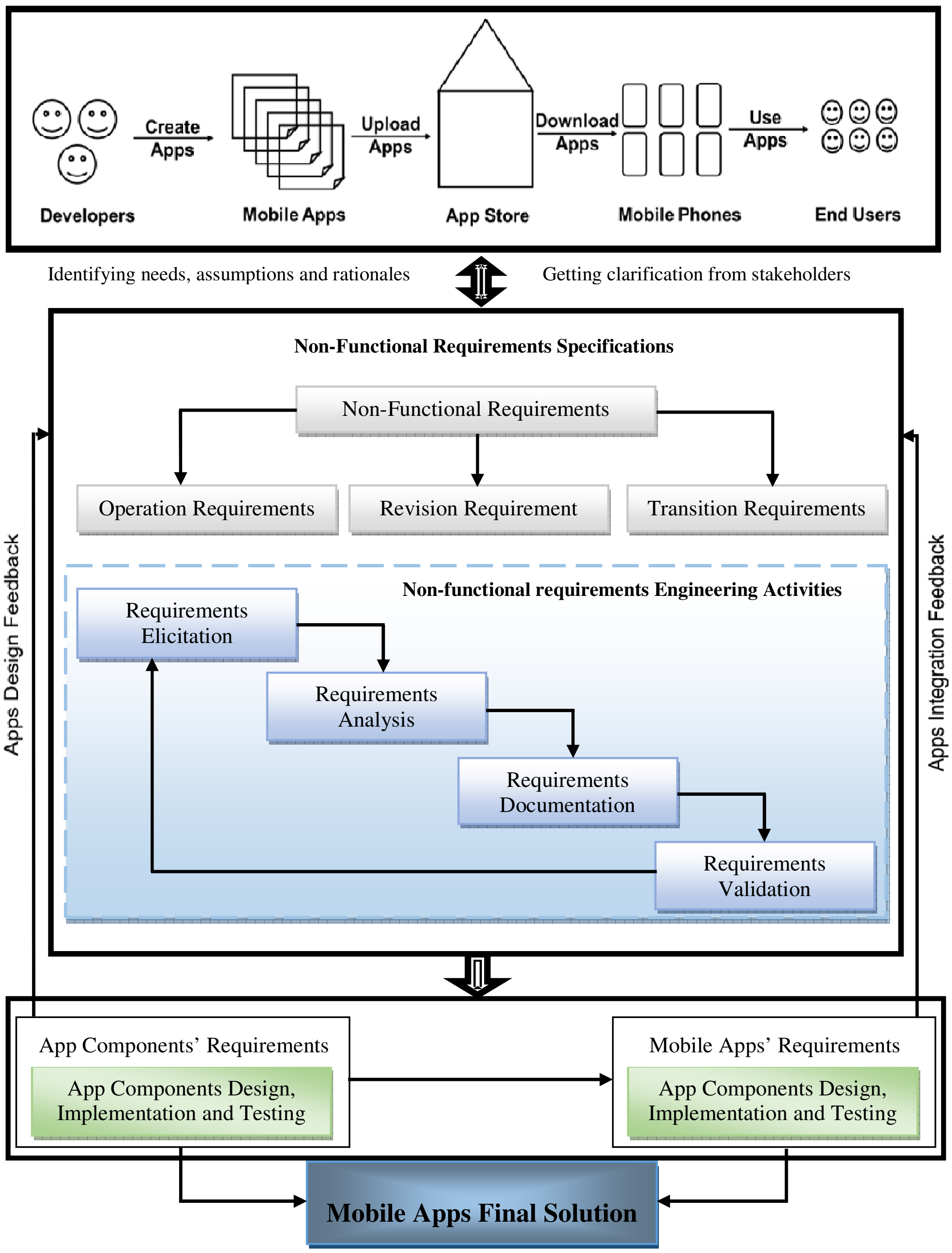
Create Apps
Upload Apps
Download Apps
Use Apps
Developers
Mobile Apps
App Store
Mobile Phones
End Users
Identifying needs, assumptions and rationales
Getting clarification from stakeholders
Non-Functional Requirements Specifications
Non-Functional Requirements
Operation Requirements
Revision Requirement
Transition Requirements
Non-functional requirements Engineering Activities
Requirements Elicitation
Requirements Analysis
Requirements Documentation
Requirements Validation
Apps Design Feedback
Apps Integration Feedback
App Components' Requirements
App Components Design, Implementation and Testing
Mobile Apps' Requirements
App Components Design, Implementation and Testing
Mobile Apps Final Solution

Figure 1. The proposed data-driven model for NFR in MAD

3.1. NON-FUNCTIONAL REQUIREMENTS ELICITATION

Elicitation is about collecting the requirements from stakeholders. The upper part of this model identifies stakeholders as the source of data. Though the conventional methods of requirement gathering such as questionnaire and interview are still important, identifying and capturing non-functional requirements from mobile-app users can be a quite complicated process as mobile-app users find it difficult to specify these hidden requirements. Therefore, a wide range of information on non-functional requirements can be mined from the application markets, the app rating and review where mobile app users can depict their assessments on the mobile app can be used as a source of data.

Subsequently, such information can be turned into a source where non-functional requirements can be extracted by mobile-app developers. The information is rich in what mobile-app user's need from the application in terms of functionality and solutions to bug-related problems, alongside praise for the features they cherish. In this model, the target problem and the stakeholders' needs should be identified and mine by the mobile app developer, in order to look for rationales and assumptions from the identified needs representing the basis for an appropriate requirements specification.

3.1.1 DATA SOURCES AND EXTRACTION TECHNIQUES

Various examinations have utilized Stack Overflow information to classify its inquiries and dissect textual contents of discourses [16], One of the principal discoveries is that designers vigorously depend on Q&A sites such as Stack Overflow, Programmers Exchange, Project Management and Quora for valuable information, however, Stack Overflow is preferred because significant number of the inquiries get addressed rapidly.

Firstly, this research adopts [21] approach for data from Stack Overflow. This approach comprises of three key strides. Initially, the post and comments are extracted from Stack Overflow followed by data refinement and tokenization (pre-processing). In addition, we develop a subject model LDA to outline the subject of the corpus. At last, we mark the subjects with the NFRs by our wordlists. This enables us to talk about the implication of dealing with NFRs in mobile application development. This work considered only a part of the NFR that we clearly identified as a desired system property and dismissed whatever remains of the NFR (e.g., because of unessential/ambiguous data).

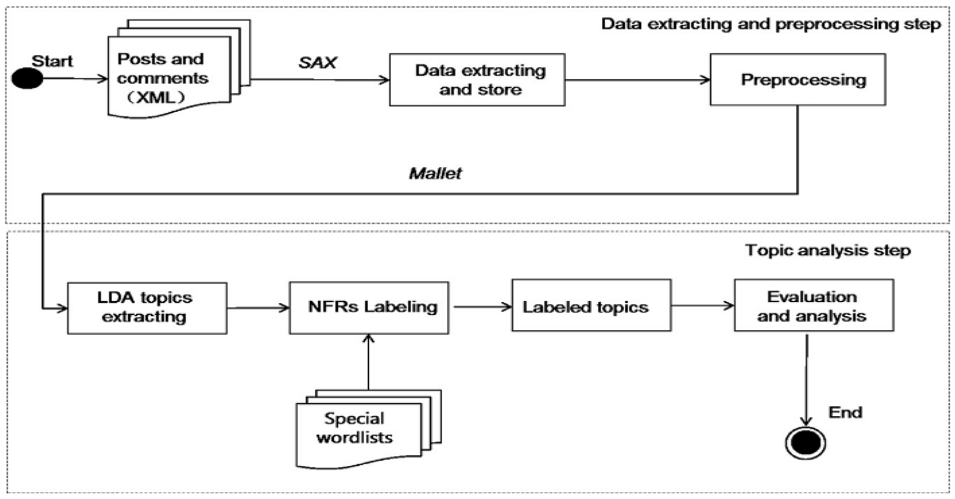

Figure 2. Data extraction approach for Q&A websites

Secondly, this research use AR-Miner (a novel computational structure for App Review Mining by performs thorough investigation from rudimentary user-reviews) for data form app marketplaces, such as Apple App Store and Google Play, to encourage mobile application engineers to find the most "constructive" user-reviews from an expansive and quickly expanding the pool of user-reviews. This approach comprises of four key strides (i) first mining constructive user-reviews by filtering relevant and irrelevant reviews, (ii) then grouping the constructive user-reviews naturally utilizing subject modelling, (iii) additionally organizing the useful user-reviews by a successful review ranking system, (iv) lastly introducing the collection of most "constructive" user-reviews by means of an instinctive perception approach.

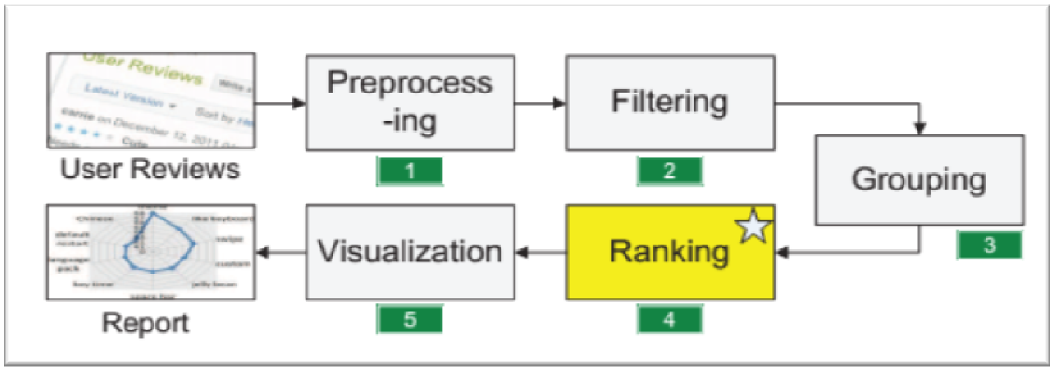

Figure 3. Data extraction approach for app marketplaces (App Store and Google Play)

3.1.2 Latent Dirichlet Allocation-Based Subject Modelling

LDA has been recognized as one of the best techniques for finding the topics of discussions in natural language text documents. It has been applied to various software engineerings research questions, such as requirement gathering, software defect prediction, bug localization, and software change message classification. In LDA, the topic is the conditional probability distribution of words in the vocabulary. It uses word frequencies and co-occurrences of frequencies in the document to build a model of related words. That is, LDA creates topics when it finds there are sets of words that tend to co-occur frequently in the documents of the corpus. The words in a topic are usually semantically related.

To help understand the Stack Overflow discussions, we apply the topic model LDA to summarize the topics of the corpus. For example, a topic that contains the words “reliability, failure, error, redundancy, fails, bug, crash, stable, reliable, maturity, recoverability, fault tolerance” (because these words occur together frequently in documents of the corpus), indicates that this topic is related to reliability.

3.2. Non-Functional Requirement Analysis, Documentation & Validation

The middle part of this model identifies and discussed the process of non-functional requirement engineering activities based on mobile-apps quality factors and design criteria. The taxonomy of quality attributes [19] is adopted due to the fact that it is the modified version of [9] software quality measurement manual, it also contained procedures and guidelines for assisting software system developers in setting quality goals, applying metrics and making quality assessments. The efficiency of this modified model plus its alignment with mobile-app development makes it perfect for dealing with the non-functional requirement in mobile-apps development.

During Analysis of the non-functional requirements, identifying the relative significance of every quality factor from the user's perspective and additionally recognizing the outline criteria on which these elements depend is as imperative as making requirements quantifiable. The quality factor such as reliability can be measured using the mean time to failure which can be tested by running the app and count crashes per hour. Taxonomy of quality attributes [19] is used as a base for identifying user concerns with the mobile app as shown in Table 1 below.

Table 1. User concerns based on a taxonomy of quality attributes.

User Needs	User concerns with the Mobile App	Non-Functional Category
Operation How well does the system perform for daily use?	Can I run it?	Usability (USB)
	Is it secure?	Integrity (INT)
	Will it run on my hardware as well as it can?	Efficiency (EFC)
	Does it do it accurately all the time?	Reliability (REL)
	Does it do what I want?	Correctness (CRT)
Revision How easy is it to correct errors and add on functions?	Can I change it?	Flexibility (FLX)
	Can I fix it?	Maintainability (MNT)
	Can I test it?	Testability (TST)
Transition How easy is it to adapt to changes in the technical environment?	Will I be able to interface it with another app	Interoperability (IOP)
	Will I be able to use it on another machine?	Portability (POR)
	Will I be able to reuse some of the apps	Reusability (REU)

In the early mobile app development stages, each stakeholder has different needs that must be considered. The proposed model ensures the identification and documentation of stakeholder's needs and their documentation sources. It was noteworthy that there was a direct link between the stakeholders and the needs, and also between the documentation sources and the needs. User feedback includes a variety of information. Users may share thoughts on the best way to enhance by adding or changing features. This feedback helps provide documentation of the application, its requirements, and features. The INVEST principle (independent, negotiable, valuable, estimable, Small, Testable) Can Be Used To Write Detailed User Stories And For Each User Story, A Scenario, acceptance criteria, limitations, and constraints can be written to avoid developers guessing the details for a given feature during implementation.

A good mobile application' requirement specification must be correct, complete, unambiguous, consistent, ranked for importance and stability, modifiable, traceable. Inspection is the primary way to validate non-functional requirements and all the stakeholders mentioned in the elicitation process must be involved. A validation checklist can be used in getting clarification from stakeholders based on the questions raised in Table 1.

3.3. Back End (Modelling Layer)

Once finished the components verification, all activities of the third process of "System Integration and Testing" were conducted taking into account all systems' components. At this point, it is recommended to send feedback to the previous processes, aiming to improve system requirements' specification throughout this iterative and incremental process.

Component-based architecture can be used to design and facilitate the system in a stage by stage process with the involvement of the stakeholders. This can be a very tedious and time-consuming task but it's a worthwhile process. The advantage of approaching the developed system in a stage by stage basis with the feedback of the users will enable the team to tackle and address all the issues at the initial stages itself, thus avoiding disappointments after the final application is delivered.

The proposed model uses product-oriented approaches that focus on apps quality, capture operational criteria for each non-functional requirement so that it can be measured once the application is built. Furthermore, it's important to note that quality factors and design criteria are related, each factor depends on a number of associated criteria, e. g. correctness depends on completeness, consistency, traceability while verifiability depends on modularity, self-descriptiveness, and simplicity.

4. Empirical Evaluation And Results

To evaluate if the proposed model can really help and support application developers in dealing with the ever-eluding non-functional requirements in requirement engineering for mobile application development using a data-driven approach, we conduct numerous experimental studies. Specifically, we aim to answer the following questions: (1) How can the proposed model facilitate the management of NFRs just like functional requirements? (2) Why some NFRs generate more user feedback than others? (3) What are the advantages of the proposed model over conventional ways of dealing with NFRs?

We used the posts and comments of the Q&A site Stack Overflow from August 1, 2016, to July 1, 2017, to explore the NFRs trends in all discussions associated with mobile application development. Since the original data is organized in the form of an XML file containing a lot of redundant information, we used the Online XML Viewer to represent the XML data in convenient way and to extract the "title" and "body" of the posts and the "text" of the comments, totaling about 603,505 posts and 998,566 comments. For each post, we extract the tags associated with that post. Tags are keywords that users attribute with their posts.

We also use the AR-Miner to extract data from app marketplaces, such as the Apple App Store and Google Play totaling 88,591. In view of the fact that the user-reviews contain spams and unstructured information, the AR-miner helps in filtering the constructive user-reviews from the irrelevant or unconstructive user-reviews as shown in table 2. Fig. 4 shows the detailed data for each month (period), with the month as the x-axis, for example, Aug-16 means August 2016, and the number of posts, comments, and user-review as the y-axis.

Table 2. Constructive & unconstructive user-review for mobile application developers.

Class	Type (Rule)	Real Example
Constructive User-review	Performance flaw that degrades the apps' performance	It's so slow and doesn't respond to my touch, sometimes I have to restart
	Request to remove permission	This game contains too many unnecessary permissions. So annoying
	The functional flaw that produces unexpected results	None of the pictures will load in my news feed.
Unconstructive User-review	Description of apps, features, actions etc.	I have changed my review from 3 starts to 2 start
	Pure user emotional expression	This app is crap or this app is awesome
	Unclear expression of failure and question	Bad app, this is not working on my phone

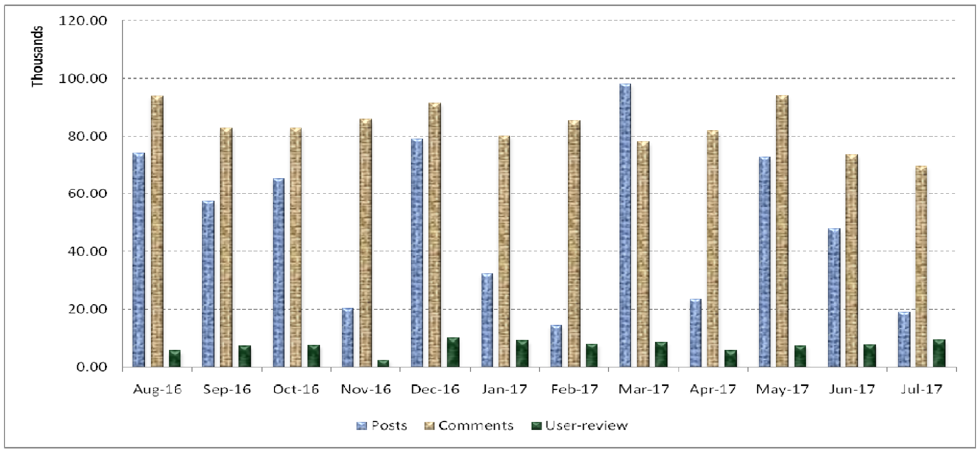

Figure 4. The number of posts, comments, and user-review

In order to determine which of our tags are relevant to NFRs related posts exclusively, we use a tag relevance threshold (TRT) value. The TRT is measured as $TRT_{tag} = \frac{No.\ of\ \ NFRs\ \ posts}{Total\ \ no.\ of\ \ posts}$ where the No. of NFRs posts is the number of posts that contained at least one of the initial set of NFRs keywords (from Table 1) and the Total no. posts are the total number of posts related to the tag. We experimented with different TRT values, manually examining the output each time, and found that using 45% yields good results without being too restrictive.

Another threshold, known as tag significance threshold (TST) is used to weed-out cases were the TRT_{tag} is = 1because incorporating such a tag is not very useful. The TST is measured as $TRT_{tag} = \frac{No.\ of\ \ NFRs\ \ posts}{No.\ of\ \ NFRs\ \ posts\ \ for\ \ the\ \ most\ \ popular\ \ tags}$.

Table 3. The NFRs and their associated wordlists based.

Labels	Related terms
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
Integrity	Security, confidentiality, integrity, accountability, authenticity, compliance, non-repudiation, secure, vulnerability, vulnerable, trustworthy, malicious, secured, exploit, compliant, access permissions, timeouts, encryption
Efficiency	efficiency, optimization, fast, slow, faster, slower, penalty, factor, sluggish, optimize, profiled, performance, efficiency, efficient, “time behavior”, “resource behavior”
Reliability	reliability, failure, error, redundancy, fail, bug, crash, stable, stability, integrity, resilience, dependability, dependableness, reliability, reliableness, responsibility, responsibleness, dependable, reliable, maturity, recoverability, “fault tolerance”
Correctness	correctness, accuracy, precision error, serviceable, serviceability, serviceableness, conformance, consistency, operability, functionality, vulnerability, secure, accurate, vulnerability, trustworthy, policy, simplicity, stability, compliant, functionality, practicality, functional, suitability, interoperability, accuracy, compliance
Flexibility	flexibility, modifiability, reconfigurability, adaptability, adjustability, changeability,

Maintainability	maintainability, modular, decentralized, encapsulation, dependency, interdependent, understandability, modifiability, modularity, maintainable, maintain, stability, analyzability, changeability, testability
Testability	testability, traceability, susceptibility, debuggability, auditability, integrity, accountability, authenticity, compliance
Interoperability	interoperability, installability, integratability, integrity, suitability, supportability, survivability, susceptibility, sustainability, transferability
Portability	portability, transferability, interoperability, documentation, internationalization, i18n, localization, l10n, standardized, migration, specification, portability, movability, movableness, portable, installability, replaceability, adaptability, conformance
Reusability	reusability, replaceability, replicability, reconfigurability, extendability, enhanceability, evolvability, expandability, adaptability, adjustability, changeability

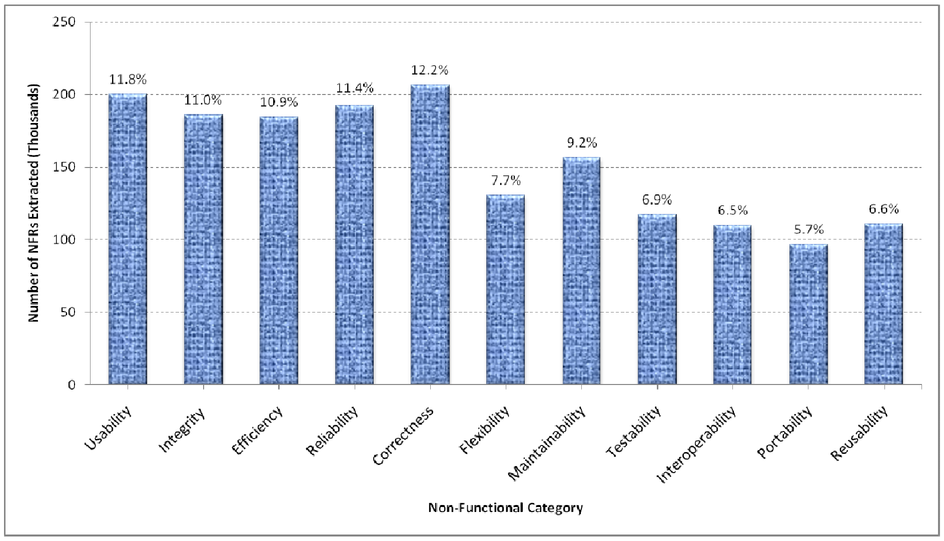

Figure 5. The number (count) and percentage of NFRs

Figure 5 shows the distribution with respect to the taxonomy of quality attributes [19] and the number (count) and percentage of NFRs (relative to the total number of NFRs) for each quality characteristic.

RQ1: How can the proposed model facilitate the management of NFRs just like functional requirements?

As shown in figure 5, the five quality attributes (correctness, usability, reliability, integrity, and efficiency) stand out with an aggregate total of more than 55%. These attributes can be defined as the ability of the application to fulfill users' objectives effectively, accurately, honorably and consistently. This essentially corresponds to a classical understanding of a functional requirement. Furthermore, NFRs at a high level typically lead to functions at lower. For example, a performance requirement may lead to throughput and subsequently functions such as transactions per second which impact specific task as shown in Figure 6. We see these results together with the abundant number of metrics for quantification as a strong argument for the proposed model can facilitate the management of NFRs just like functional requirements. Based on our data, most "non-functional" requirements describe functional aspects of an application and are, therefore, fundamentally not non-functional. From experimental observations, these indicate that the majority of NFRs can be elicited, specified, and analyzed similarly to functional requirements.

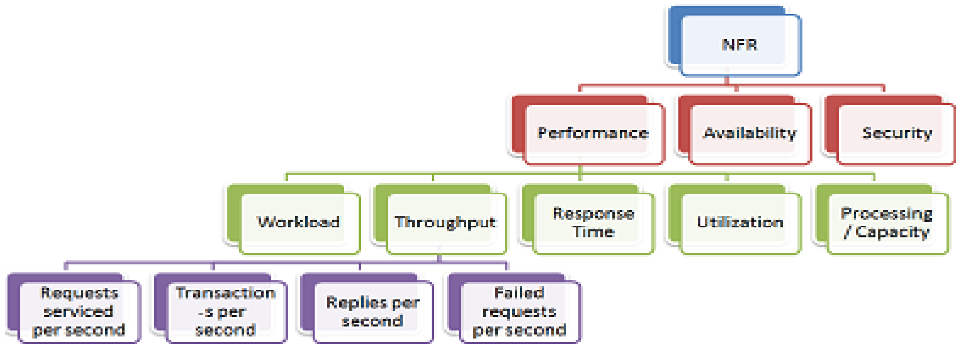

Figure 6. The Non-functional requirements hierarchy

RQ2: Why some NFRs generate more user feedback than others?

Interestingly, quality attributes such as correctness, usability, reliability, integrity, efficiency, and maintainability get a higher percentage of the user feedback. Contrarily, flexibility, testability, interoperability, portability, and reusability get the lower percentage of the stakeholders as shown in Figure 7. This indicates that most stakeholders tend to discuss NFRs that actually describe behavioral properties of application rather than NFRs that describe representational properties of the application. It also suggests the mobile application developers should attach greater importance to the correctness, usability, reliability, integrity, efficiency, and maintainability of the application.

RQ3: What are the advantages of the proposed model over conventional ways of dealing with NFRs?

Given that there is no universal definition of NFR categories, we have observed that certain discrepancy exists between the amount of NFRs information obtain using proposed model and conventional ways of dealing with NFRs as shown in Figure 7. The proposed model ensures the identification and documentation of stakeholder's needs and their documentations sources. It was noteworthy that there was a direct link between the stakeholders and the needs, and also between the documentation sources and the needs as shown in Figure 1. The proposed model uses product-oriented approaches that focus on apps quality, capture operational criteria for each non-functional requirement so that it can be measure once the application is built.

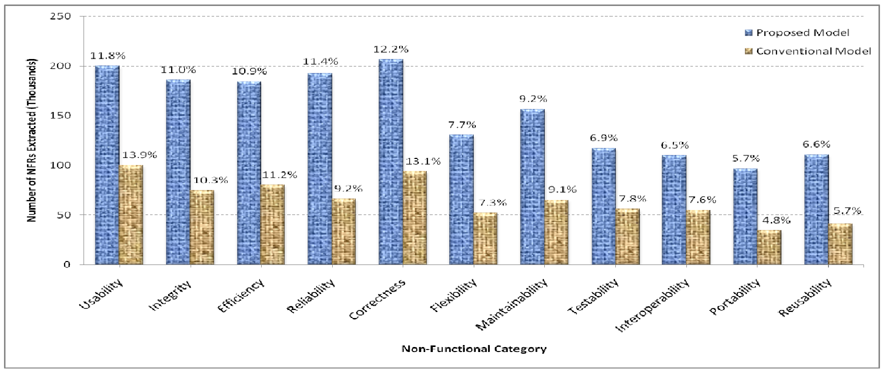

Figure 7. Number of NFRs extracted by the proposed model as compared to the conventional model

5. Conclusion And Future Work

This study set out to provide support for mobile application developers in dealing with non-functional requirements for mobile application development using a data-driven approach, the study also tries to find out if NFR can be treated the same way as functional requirements. The results of this investigation show that the proposed model can facilitate the management of NFRs just like functional requirements. Thereby enhancing the quality of requirements' engineering activities including requirements priority and traceability for mobile application development. Based on data, most "non-functional" requirements describe functional aspects of an application and are, therefore, fundamentally not non-functional. From experimental observations, these indicate that the majority of NFRs can be elicited, specified, and analyzed similarly to functional requirements. Although this study focuses on finding out if NFR can be treated the same way as functional requirements in mobile application development, the findings may well have a bearing on how mobile application developers prioritized NFRs. Given that most of the stakeholders tend to discuss NFRs that actually describe behavioral properties of application such as correctness, usability, reliability, integrity, efficiency, and maintainability rather than NFRs that describe representational properties of the application such as flexibility, testability, interoperability, portability, and reusability. Future research should, therefore, concentrate on building natural language processing (NLP) techniques that can handle limitless user-reviews, coupled with sampling techniques that take the sampling bias into account for more comprehensive user-reviews. Besides, more research is also needed to come up with a new ranking algorithm to surface apps that exhibit the best performance, lower number of crashes, and the high stability rather than just the most install or highest rating.

References

[1] Carreño, L. V. G., & Winbladh, K. (2013, May). Analysis of user comments: an approach for software requirements evolution. In Software Engineering (ICSE), 2013 35th International Conference on (pp. 582-591). IEEE.

[2] Chen, N., Lin, J., Hoi, S. C., Xiao, X., & Zhang, B. (2014, May). AR-miner: mining informative reviews for developers from mobile app marketplace. In Proceedings of the 36th International Conference on Software Engineering (pp. 767-778). ACM.

[3] Eckhardt, J., Vogelsang, A., & Fernández, D. M. (2016, May). Are" Non-functional" Requirements really Non-functional? An Investigation of Non-functional Requirements in Practice. In Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on (pp. 832-842). IEEE.

[4] Finkelstein, A., Harman, M., Jia, Y., Sarro, F., & Zhang, Y. (2013). Mining app stores: Extracting technical, business and customer rating information for analysis and prediction. RN, 13, 21.

[5] Guzman, E., & Maalej, W. (2014, August). How do users like this feature? a fine-grained sentiment analysis of app reviews. In Requirements Engineering Conference (RE), 2014 IEEE 22nd International (pp. 153-162). IEEE.

[6] Grønli, T. M., & Ghinea, G. (2016, January). Meeting quality standards for mobile application development in businesses: A framework for cross-platform testing. In System Sciences (HICSS), 2016 49th Hawaii International Conference on (pp. 5711-5720). IEEE.

[7] Huang, J., Xu, Q., Tiwana, B., Mao, Z. M., Zhang, M., & Bahl, P. (2010, June). Anatomizing application performance differences on smartphones. In Proceedings of the 8th international conference on Mobile systems, applications, and services (pp. 165-178). ACM.

[8] Iacob, C., & Harrison, R. (2013, May). Retrieving and analyzing mobile apps feature requests from online reviews. In Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on (pp. 41-44). IEEE.

[9] MacCall, J. A., & Matsumoto, M. T. (1980). Software quality measurement manual. National Technical Information Service.

[10] Malkawi, M. I. (2013). The art of software systems development: Reliability, Availability, Maintainability, Performance (RAMP). Human-Centric Computing and Information Sciences, 3(1), 22.

[11] Mushtaq, Z., Kirmani, M., & Saif, S. M. (2016). Mobile Application Development: Issues and Challenges. International Research Journal of Engineering and Technology on (pp. 1096-1099). IRJET

[12] Nagappan, M., & Shihab, E. (2016, March). Future trends in software engineering research for mobile apps. In Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on (Vol. 5, pp. 21-32). IEEE.

[13] Niemelä, E., & Latvakoski, J. (2004, October). Survey of requirements and solutions for ubiquitous software. In Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia (pp. 71-78). ACM.

[14] Rein, A. D., & Münch, J. (2013). Feature prioritization based on mock-purchase: A mobile case study. In Lean Enterprise Software and Systems (pp. 165-179). Springer, Berlin, Heidelberg.

[15] Romano, B. L., Gomes, P. M. E., Fernandes, H. C., Montini, D. Á., Dias, L. A. V., & da Cunha, A. M. (2011). A Model-Driven Method for Documentation and Analysis of Software-Intensive Systems Requirements.

[16] Rosen, C., & Shihab, E. (2016). What are mobile developers asking about? a large-scale study using stack overflow. Empirical Software Engineering, 21(3), 1192-1223.

[17] Sarro, F., Al-Subaihin, A. A., Harman, M., Jia, Y., Martin, W., & Zhang, Y. (2015, August). Feature lifecycles as they spread, migrate, remain, and die in app stores. In Requirements Engineering Conference (RE), 2015 IEEE 23rd International (pp. 76-85). IEEE.

[18] Spriestersbach, A., & Springer, T. (2004, April). Quality attributes in mobile web application development. In International Conference on Product Focused Software Process Improvement (pp. 120-130). Springer Berlin Heidelberg.

[19] Van Vliet, H. (2000). Software Engineering: Principles and Practice. New York: Wiley.

[20] Zhao, H., Qiu, M., Gai, K., Li, J., & He, X. (2015, November). Maintainable mobile model using pre-cache technology for high performance android system. In Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on (pp. 175-180). IEEE.

[21] Zou, J., Xu, L., Yang, M., Zhang, X., & Yang, D. (2017). Towards comprehending the non-functional requirements through Developers' eyes: An exploration of Stack Overflow using topic analysis. Information and Software Technology, 84, 19-32.