What Aspects do Non-functional Requirements in App User Reviews Describe?: An Exploratory and Comparative Study

Tianlu Wang
School of Computer Science
Wuhan University
Wuhan, China
wangtianlu@whu.edu.cn

Peng Liang*
School of Computer Science
Wuhan University
Wuhan, China
liangp@whu.edu.cn

Mengmeng Lu
School of Computer Science
Wuhan University
Wuhan, China
lumengmeng@whu.edu.cn

***Abstract*—App user reviews have been regarded as a valuable source to elicit user requirements. Existing research on eliciting requirements from user reviews mainly focuses on functional aspects. However, Non-Functional Requirements (NFRs) from diverse sources play a critical role during software development. In this paper, we conducted an exploratory and comparative study in order to gain a deeper understanding on the nature of NFRs in user reviews, and to further compare the difference on the distributions of various NFRs between user reviews and industrial requirements specifications with respect to the system properties that NFRs address. We used 1278 user review sentences from two popular Apps (iBooks and WhatsApp) classified as NFRs that address quality characteristics, and further classified these 1278 NFRs based on the *system view* and the *behavior theory* and compared the difference on the distributions of various NFRs between user reviews and industrial requirements specifications. The results show that in App user reviews, users primarily report quality aspects on *Reliability* and *Usability*, and over 50% NFRs address *interface* behavior of the systems. Overall the distributions of NFRs with respect to the *system view* and the *behavior theory* in user reviews and industrial requirements specifications are similar, but the distributions of NFRs classified as *architecture* and *state*, and the distributions of NFRs in certain quality characteristic classes (e.g., the *Usability* class) show some differences. We concluded that most NFRs in user reviews are essentially not non-functional since they describe behavior over the interface of the systems. Compared with NFRs in industrial requirements specifications, NFRs in user reviews report more state-related aspects of the systems and care less about the architecture of the systems.**

Keywords*—*Crowd-based Requirements Engineering, App User Reviews, Classification of NFRs, Comparative Study

I. INTRODUCTION

User reviews have been regarded as a valuable source for eliciting user requirements to design future software releases [1]. In modern app marketplaces (e.g., Apple Store and Google Play), users can not only give star rating on the Apps, but also post reviews about feature requests, bug reports, and their experience when using the apps [2][3]. User reviews that contain information about user requirements can be classified and analyzed according to the content, which will help developers improve software quality and identify missing features [4].

Normally requirements are classified as either Functional Requirements (FRs) or Non-Functional Requirements (NFRs) [5]. Roughly speaking, FRs describe what the system should do and NFRs place constraints on how these FRs are implemented [6]. NFRs are widely recognized to play a critical role during software development. However, recent research shows that they have received little attention, and are often poorly understood and not considered adequately during software development [7][8]. For example, NFRs are not elicited at the same level of details as FRs, and they are not rigorously articulated in requirement documents [8][9].

Broy proposed a new view to rethink the nature of NFRs. According to the properties of the systems that NFRs addressed, he divided NFRs into two types: *behavioral* NFRs and *representational* NFRs, and *behavioral* NFRs are further classified as *interface*, *architecture*, and *state* [10][11]. Eckhardt *et al*. classified NFRs in industrial requirements specifications based on this view. They found that about 51.5% NFRs describe *interface* behavior of the systems. Since FRs are known to describe behavior over the interface of the systems, the authors concluded that most "non-functional" requirements are basically not non-functional because these NFRs describe the same type of behavior as FRs do [12].

Aside from industrial requirements specifications, App user reviews is another source to describe requirements. Research on eliciting user requirements from App user reviews has focused on functional aspects. Yet the quality of an App is vital for its success and NFRs in user reviews that are related to quality characteristics should be considered as well [13]. In our previous work [14], we classified App user reviews according to the quality characteristics defined in ISO/IEC 25010 [15], and the results show that among 4000 randomly collected user review sentences, 1278 sentences (about one third) addressed the quality aspects. Such a proportion also denotes that NFRs should demand more attention in App development.

In this paper, in order to gain a deeper understanding on the nature of NFRs in App user reviews, we conducted a manual classification on 1278 user review sentences also according to the view proposed by Broy with respect to the *system view* and the *behavior theory* [10][11]. The 1278 sentences used in this work are sentences classified as NFRs in our previous work [14]. The results show that about 54.4% NFRs describe *interface* behavior of the systems. Hence, we can conclude that most NFRs in App user reviews are essentially not non-functional because they describe the same type of behavior as FRs do. We further classified *behavioral* NFRs based on the *behavior theory* they used to express the desired properties of the systems. We also compared the

* Corresponding author
This work is sponsored by the NSFC under Grant No. 61472286.

978-1-7281-1970-0/18/$31.00 ©2018 IEEE
DOI 10.1109/APSEC.2018.00064

classification results of NFRs in App user reviews and industrial requirements specifications in [12]. We found that overall the distributions of NFRs are similar, however, the distributions of NFRs classified as *architecture* and *state*, and the distributions of NFRs in certain quality characteristics classes show some differences in the two sources.

The contribution of this paper is twofold. First, we classified NFRs in App user reviews based on the *system view* and the *behavior theory* to gain a deeper understanding to the nature of NFRs. Second, we compared our classification results of NFRs in App user reviews with those of NFRs in industrial requirements specifications presented by Eckhardt *et al*. [12] to explore the difference of NFRs between the two sources.

The remainder of the paper is organized as follows. Section II introduces the background and related work. Section III describes the research questions. Section IV describes the study design including the dataset, classification schemes, classification process, and data analysis process. Section V reports on the study results and then we discussed the results and compared our classification results with those in another study in Section VI. Section VII introduces the threat to validity and Section VIII concludes this study and gives the prospect of the future work.

II. Background and Related Work

A. Crowd-based Requirements Engineering and User Feedback

Crowd-based Requirements Engineering (CrowdRE) is an umbrella term proposed by Groen *et al*. [16] for automated or semi-automated RE approaches to obtain and analyze information from a crowd to derive validated user requirements. In most cases, the crowd in CrowdRE is a large group of current and potential users of a software product who interact among themselves or with representatives of a software company (e.g., the product owner or development team) [17]. User feedback provided in App stores in the form of reviews can be analyzed to derive user requirements in order to improve software quality and identify missing features, which is a typical case of CrowdRE. Pagano and Maalej analyzed over one million reviews from the Apple AppStore to investigate how and when users provide feedback, identify and classify topics in user reviews, and analyze the impact of feedback content on the user community. They found that reviews typically contain multiple topics (e.g., user experience, bug reports, and feature requests) and further classified these topics into four themes: community, requirements, rating, and user experience [18]. In this study, we focused on the requirements in App user reviews that are related to quality characteristics.

User reviews can be classified either automatically or manually. Maalej and Nabil introduced several classification techniques to automatically classify user reviews into bug reports, feature requests, user experiences, and ratings. The authors compared the accuracy of these classification techniques (simple string matching, text classification, natural language processing, sentiment analysis, and review metadata) and they found that multiple binary classifiers outperformed single multiclass classifiers [19]. McIlroy *et al*. proposed an approach that can automatically assign multi-labels to user reviews based on the raised issues (e.g., feature requests, functional complaints, and privacy issues). The proposed approach can help various stakeholders gain an overview of the users' feedback that is readily available in the reviews, provide an overview of an App store, and allow for the detection of anomalous Apps [20]. Groen *et al*. manually tagged online reviews about the Apps according to ISO/IEC 25010 [15] and then defined 16 language patterns by assessing and comparing the statements from the reviews tagged as *Usability*. They performed a tool-based pattern matching with the 16 language patterns in the complete dataset, and the result shows that the proposed language patterns can help them identify and classify user reviews on *Usability* automatically and efficiently [13].

We have also made effort on automatically classifying user reviews. In our previous work [14], we manually classified 4000 user review sentences from iBooks and WhatsApp (2000 sentences for each App) into seven types: *Reliability*, *Usability*, *Performance*, *Portability*, *Security*, *FR*, and *Others* as the benchmark dataset. We then proposed a classification technique AUR-BoW and compared the F-measures of the classification results through all the combinations of four classification techniques (including our proposed technique) with three machine learning algorithms to classify user reviews. In this study, we conducted a manual classification on user reviews and used the 1278 sentences classified as NFRs in our previous work [14] as the dataset (see Section IV-A).

B. Classifications of Non-functional Requirements

The requirement engineering community has classified requirements into Functional Requirements (FRs) and Non-Functional Requirements (NFRs). Even though there is still no consensus on the definition of NFRs, there has been considerable work characterizing and classifying NFRs [7]. An early work on software quality [21] presented a tree of software quality characteristics, in which meeting a parent quality implies meeting its offspring qualities. The tree includes a number of "-ilities" for a variety of quality attributes. Roman classified NFRs into six classes: *Interface* , *Performance*, *Operating*, *Lifecycle*, *Economic*, and *Political* [22]. Glinz raised the classification problems of requirements into FRs and NFRs, and presented a new classification of requirements based on four facets: *kind* (e.g., function, performance, or constraint), *representation* (e.g., operational, quantitative, or qualitative), *satisfaction* (e.g., hard or soft), and *role* (e.g., prescriptive or assumptive), which can overcome the fuzziness of the traditional classification into FRs and NFRs [23].

Quality is the most popular term adopted to specify NFRs and ISO/IEC 25010 [15] is the international standard to classify software qualities [18]. It defines a product quality model that categorizes software product quality into eight quality characteristics (*Functional suitability*, *Performance efficiency*, *Compatibility*, *Usability*, *Reliability*, *Security*, *Maintainability*, and *Portability*) and each quality characteristic is composed of a set of sub-characteristics. In this study, we used ISO/IEC 25010 as one of the classification schemes for NFRs in App user reviews.

According to the view proposed by Broy [10][11], Eckhardt *et al*. [12] classified 530 NFRs extracted from 11 industrial requirements specifications with respect to the *system view* and the *behavior theory* in order to increase the

understanding on the nature of the NFRs that address system properties. Their results suggest that most NFRs actually describe the behavior of the systems and thus are basically not non-functional. Therefore, many so called NFRs should be handled similar as FRs. In our study, we also focused on the system properties that NFRs address and employed the same classification schemes [10][11] to classify NFRs in App user reviews. Additionally, we compared the results of our study with the results of [12] in order to increase our understanding on the nature of NFRs and learn the difference on the distributions of various NFRs between App user reviews and industrial requirements specifications.

III. Research Questions

The goal of this study is to gain a deeper understanding on the nature of NFRs in App user reviews with respect to the *system view* and the *behavior theory*. Furthermore, we want to compare the difference on the distributions of various NFRs between App user reviews and industrial requirements specifications from different aspects, including the distribution of NFRs based on quality characteristics, the type of system behavior NFRs address, and the *behavior theory* that NFRs use. In order to achieve our goal and make the results between App user reviews (this study) and industrial requirements specifications ([12]) comparable, we formulate four Research Questions (RQs) which are the same as the RQs of [12] except for the RQ on application domains, and the comparison results are presented in Section VI-B:

RQ1: What is the distribution of NFR classes in App user reviews?

Rationale: With this RQ, we want to get an overview of NFR classes with respect to the quality characteristics users reported in App user reviews.

RQ2: How many NFRs describe system behavior?

Rationale: With this RQ, we want to know how many NFRs describe system behavior (the remaining NFRs describe the representation of a system). Moreover, we want to compare the proportions of *behavioral* NFRs and *representationa*l NFRs in different NFR classes.

RQ3: Which *system views* do behavioral NFRs address?

Rationale: With this RQ, we want to identify the system views that NFRs address and the types of system behavior that *behavioral* NFRs describe, and learn more about the relation between the system views and NFR classes.

RQ4: In which type of *behavior theory* are behavioral NFRs expressed?

Rationale: With this RQ, we want to identify the behavior theories used to express *behavioral* NFRs and learn more about the relation between behavioral theories and NFR classes.

IV. Study Design

Considering the exploratory nature of the research questions as well as the comparison planned, we conducted an exploratory and comparative study [25]. In this study, we classified 1278 NFRs in App user reviews based on the *system view* and the *behavior theory*, and compared the distribution of these 1278 NFRs with the distribution of 530 NFRs extracted from 11 industrial requirements specifications in [12]. The industrial requirements specifications are from five different companies for six application domains (Finance, Automotive, Travel Management, Railway, Traffic Management, and Facility Management). This section details the design of this study.

A. Dataset

The dataset used in this study consists of 1278 user review sentences that come from 4000 randomly collected user review sentences of iBooks (iOS) and WhatsApp (Android) (2000 sentences from each App). In our previous work [14], according to ISO/IEC 25010 [15], of 4000 user review sentences 1259 sentences were classified into four quality characteristics classes: *Portability*, *Reliability*, *Performance efficiency*, and *Usability*, and 19 sentences were classified as *Security*. Considering that the 19 sentences about *Security* are too few (compared to 4000 sentences) for training classifiers, we excluded them in our previous work [14] that focus on automatic classification of NFRs, while in this exploratory study, we manually performed a classification and analysis on the 1278 user review sentences (including the 19 sentences classified as *Security*) and every sentence is regarded as an NFR.

B. Data Classification

1) Classification Schemes: In order to answer the RQs defined in Section III. and compare the results between our study and the study that focuses on the classification of NFRs in industrial requirements specifications [12], we classified the 1278 NFRs in App user reviews based on the following three classification schemes that are also employed in [12] (the classification results on the dataset have been provided online[1]):

- **NFR class**: In this study, NFRs are classified into different NFR classes based on the quality model presented in ISO/IEC 25010 [15], an extended standard to replace ISO/IEC 9126 [26] that was used in [12]. Each NFR class in this study represents a quality characteristic.
- **System view**: As with the classification of NFRs in industrial requirements specifications in [12], the classification of NFRs in App user reviews in this study is based on the view proposed by Broy [10][11] and we assign a *system view* to each *behavioral* NFR. According to Broy's view, NFRs can be divided into *representational* NFRs that describe properties not related to the behavior of the system (i.e., the way that a system is syntactically and technically represented) and *behavioral* NFRs that describe behavioral properties of the system. *Behavioral* NFRs are further classified into two types: *behavioral* NFRs that accord with the *black-box* view and *behavioral* NFRs that accord with the *glass-box* view. *Black-box* view is related to the *interface* behavior of a system and *glass-box* view pertains to the internal structure and internal behavior of a system. There are two categories of *glass-box* behavior: *architecture* and *state*. The description of each category is shown in TABLE I.

[1] https://tinyurl.com/y7tj9lkq

TABLE I. CLASSIFICATION OF NFRS WITH RESPECT TO THE SYSTEM VIEW

Category	System view	Description	Example
Representational	-	NFRs that describe the way a system is syntactically and technically represented	*Ugliest interface ever with ugliest emotions ever.*
Behavioral	Interface (Black-box)	NFRs that describe the behavioral properties of a system at its interface	*I have purchased some books that have not shown up in the library.*
	Architecture (Glass-box)	NFRs related to the connections and interactions of sub-systems or system components	*Synchronizing between devices is not always consistent.*
	State (Glass-box)	NFRs that describe the state space or state transitions of a system	*App is not responding while sending the pictures.*

TABLE II. CLASSIFICATION OF NFRS WITH RESPECT TO THE BEHAVIOR THEORY

Category	Description	Example
Syntactic	NFRs that are expressed by a syntactic structure on which behavior can be described	*But the images are compressed way too much.*
Logical	NFRs that are expressed by a set of interaction patterns	*Random chat opens up when I check WhatsApp aft a while.*
Timed	NFRs that are expressed by a set of interaction patterns with relation to time	*It takes near an entire day to load them on.*
Probabilistic	NFRs that are expressed by probabilities for a set of interaction patterns	*Sometimes store not connected or server not available.*
Probabilistic & Timed	NFRs that are expressed by probabilities for a set of interaction patterns with relation to time	*Sometimes if you wait a while the list of books comes up.*

- **Behavior theory**: Each NFR is described with a certain *behavior theory*. In this study, we used the categories of behavior theories in [12] to differentiate NFRs. The categories and their descriptions are shown in TABLE II.

2) Classification Process: When the dataset was prepared, all NFRs were manually classified by the three authors. In our previous work [14], 1278 NFRs had been manually classified into five quality characteristic classes according to ISO/IEC 25010 [15], which are regarded as five NFR classes (see Section IV-B-1)) in this study (note that when we conducted the classification in our previous work [14], we were not aware of the other two NFR classification schemes used in this study at that time). Based on the other two classification schemes (the *system view* and the *behavior theory*) presented in Section IV-B-1), we conducted a manual classification on 1278 NFRs in App user reviews. We first conducted a pilot classification of 40 NFRs (10 NFRs for each quality characteristic class) based on the *system view* and the *behavior theory*, and any disagreements were discussed and resolved by the three researchers. After that, we gained a consensus on the classification. Then the first author and the third author conducted the classification on the remaining NFRs independently, and any disagreements between them were discussed and confirmed with the second author. At last, we achieved an agreement on the classification of all NFRs.

3) Data Analysis: After the classification process, we conducted an analysis on the classification results. To answer RQ1, we analyzed the distribution of NFRs with respect to the quality characteristics defined in ISO/IEC 25010. To answer RQ2, we compared the numbers of *representational* NFRs and *behavioral* NFRs. To answer RQ3, we analyzed the distribution of *behavioral* NFRs that are classified as *interface*, *architecture*, and *state*. To answer RQ4, we analyzed the distribution of *behavioral* NFRs with respect to the behavior theory they used.

In addition, we compared the results in this study with the results in [12] and analyzed the difference between them. Therefore, we are able to gain a deeper understanding on the nature of NFRs in App user reviews and industrial requirements specifications as well as the importance of NFRs in these two settings.

V. STUDY RESULTS

RQ1: What is the distribution of NFR classes in App user reviews?

The result of RQ1 comes from our previous work [14], and is presented in this study for comparison. TABLE III shows the distribution of NFR classes with respect to the quality characteristics defined in ISO/IEC 25010 [15]. We can see that App users reported quality issues most frequently on *Reliability* (around 45.9%), followed by *Usability* (around 33.7%). Quality issues on *Performance efficiency* and *Portability* both account for a small proportion of NFRs in App user reviews, around 9.5% and 9.3% respectively, and quality issues on *Security* account for the least part in App user reviews (around 1.6%).

TABLE III. DISTRIBUTION OF NFR CLASSES IN APP USER REVIEWS

Quality characteristic	#Sentence	Proportion
Reliability	587	45.9%
Usability	432	33.7%
Performance efficiency	121	9.5%
Portability	119	9.3%
Security	19	1.6%
Total	**1278**	**100%**

RQ2: How many NFRs describe system behavior?

The results of RQ2 are presented in Fig. 1. The table of Fig. 1 shows the distribution of *behavioral* NFRs and *representational* NFRs in App user reviews. With regard to *behavioral* NFRs, according to the characteristic of behavior they describe, *behavioral* NFRs are further divided into two types: *black-box* and *glass-box*. The bar chart of Fig. 1 shows the percentages of NFRs that are classified as *black-box*,

glass-box, and *representational* NFRs within each quality characteristic class.

Of all 1278 NFRs around 70.3% NFRs describe the *behavioral* aspects of the system, while the rest 29.7% NFRs describe the *representational* aspects of the system.

The results show that more than half of NFRs in App user reviews describe *black-box* behavior (external behavior over the system interface). Furthermore, we can see that in the *Security*, *Portability*, *Performance efficiency*, and *Reliability* class, users mentioned most often *black-box* behavior of the system. For example, in the *Performance efficiency* class, over 97% NFRs are classified as *black-box* and most *Performance efficiency* requirements describe the time that users spent on achieving specific goals, such as "*it takes at least 30 minutes to download my books*". We also find that users usually wrote reviews qualitatively with words "*long*", "*fast*", "*slow*", "*smooth*" and other similar words to describe time characteristic instead of quantitatively with number-related words such as "*15 seconds*".

Behavioral vs. Representational	#Sentence	Proportion
Behavioral	**899**	**70.3%**
Black-box	695	54.4%
Glass-box	204	15.9%
Representational	**379**	**29.7%**

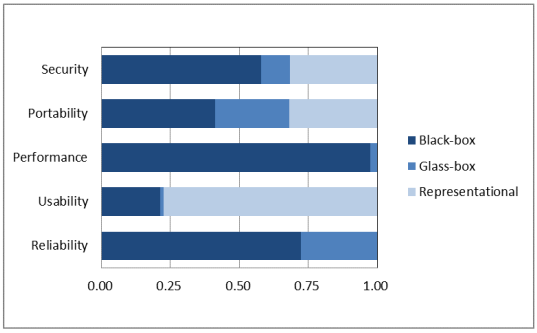

Fig. 1. Distribution of behavioral and representational NFRs.

Aside from NFRs that describe *black-box* behavior, there are also NFRs in App user reviews that describe internal structure and internal behavior of the system, and this type of NFRs are classified as *glass-box* [10][11]. One example *Portability* NFR classified as *glass-box* is "*it keeps crashing million times a day after the last update*". The results also show that NFRs classified as *glass-box* are mainly distributed in the *Portability* and *Reliability* class, and the *Performance efficiency* and *Usability* class both have a low share of NFRs on *glass-box*. In the *Performance efficiency* class, the NFRs classified as *glass-box* are about the interaction of system components with relation to time, such as "*the slowness of the iBookstore just drives me nuts*", while in the *Usability* class, the NFRs classified as *glass-box* are related to the state of the system or the data stored in the system.

We also see that *representational* NFRs appear in the *Security*, *Portability*, and *Usability* class, while in the *Performance efficiency* and *Reliability* class, no NFRs are classified as *representational*. Around 77.5% of *Usability* NFRs are classified as *representational*, which is not surprising because most NFRs in the *Usability* class are about the GUI of Apps, for example, "*ugliest interface ever with ugliest emoticons ever*". In the *Portability* class, *representational* NFRs mainly describe the adaptability of Apps to the platforms or operating systems, such as "*latest version is not compatible with iOS 6*". It is worth mentioning that in the *Portability* class, the three types *black-box*, *glass-box*, and *representational* have a relatively even distribution. While in the *Security*, *Performance efficiency*, *Usability*, and *Reliability* class, App users focus on one aspect of the system (i.e., *representational* aspects of the user interface in the *Usability* class *and black-box* behavior over the system interface in the *other four* NFR classes).

RQ3: Which system views do behavioral NFRs address?

For RQ3, We only considered *behavioral* NFRs as *representational* NFRs do not describe system behavior. Fig. 2 shows the results for RQ3. The distribution of *behavioral* NFRs classified based on the *system view* (see Section IV-B-1)) is shown in the table of Fig. 2. More precisely, the percentages of *behavioral* NFRs that are classified as *interface*, *architecture*, and *state* within each quality characteristic class are shown in the bar chart of Fig. 2.

System view	#Sentence	Proportion
Interface	695	77.3%
Architecture	28	3.1%
State	176	19.6%

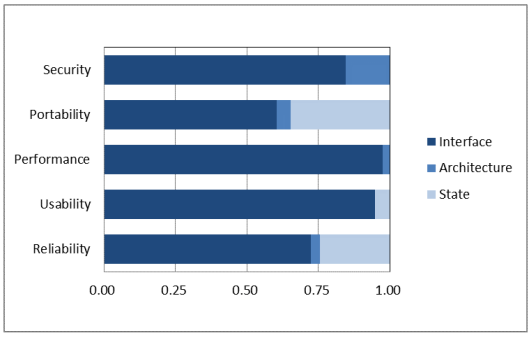

Fig. 2. Distribution of behavioral NFRs with respect to the system view.

A majority of *behavioral* NFRs are classified as *interface* (around 77.3% of all NFRs that describe behavior), in the remaining NFRs around 19.6% are classified as *state* and 3.1% are classified as *architecture*. We can see that all quality characteristic classes have a highest share of NFRs on *interface*. For example, in the *Reliability* class, users often reported the observable abnormal interface behavior of the App resulting from errors within the system, such as "*because of any bug in the app I'm unable to receive voice on speaker during voice call or video call*". While in the *Security* class, users expressed the dissatisfaction on the disclosure of their private information, such as "*I don't like sharing my personal information with Facebook*".

We can also see that all quality characteristic classes except for the *Performance efficiency* class contain *behavioral* NFRs that describe state-related behavior, and most of these behavioral NFRs are about the state transition of the system. For example in the *Reliability* class, there are many *behavioral* NFRs describing the breakdown of the system like "*App crashes every time I try to buy a book*". There are also *behavioral* NFRs about the state of the data stored in the system, such as "*when we delete all chats from setting then broadcast list also gets deleted*".

As for NFRs classified as *architecture*, they are distributed in the *Security*, *Portability*, *Performance efficiency*, and *Reliability* class with a small proportion.

There are 4 NFRs (around 4.9% of *behavioral* NFRs) about synchronization among devices in the *Portability* class, which can be considered as the interaction between the server and other system components, for example, "*however because I have bought books and though I can open them on Apple device in which they were purchased, I can't find them on my other Apple devices*". Thus we classify them as *architecture*. Furthermore, there are 3 NFRs (around 2.5% of *behavioral* NFRs) in the *Performance efficiency* class and 19 NFRs (around 3.2% of *behavioral* NFRs) in the *Reliability* class that are classified as *architecture* as they also describe the interactions among sub-systems or system components. For example the *Reliability* NFR"*I can't seem to send the books that I've read to the cloud*".

RQ4: In which type of behavior theory are behavioral NFRs expressed?

For RQ4, we also only considered *behavioral* NFRs as *representational* NFRs do not describe system behavior. The results of RQ4 are shown in Fig. 3. The table of Fig. 3 shows the distribution of *behavioral* NFRs classified based on the *behavior theory* (see Section IV-B-1)). More precisely, the bar chart of Fig. 3 shows the percentages of *behavioral* NFRs that are classified as *syntactic*, *logical*, *timed*, *probabilistic*, and *probabilistic & timed* within each quality characteristic class.

Behavior theory	#Sentence	Proportion
Syntactic	244	27.1%
Logical	463	51.5%
Timed	133	14.8%
Probabilistic	45	5.0%
Probabilistic & Timed	14	1.6%

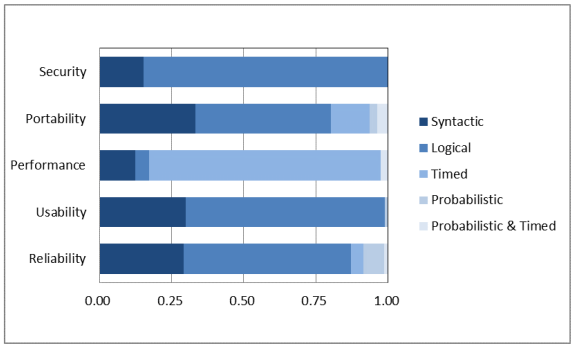

Fig. 3. Distribution of behavioral NFRs with respect to their behavior theory.

Most NFRs (around 51.5% of all *behavioral* NFRs) in App user reviews are *logical*, followed by *syntactic* (around 27.1% of all *behavioral* NFRs), *timed* (around 14.8% of all *behavioral* NFRs), *probabilistic* (around 5.0% of all *behavioral* NFRs), and *probabilistic & timed* (around 1.6% of all *behavioral* NFRs). All quality characteristic classes have a highest share of NFRs on *logical* except for the *Performance efficiency* class. In the *Performance efficiency* class, around 80.2% of *behavioral* NFRs are *timed* NFRs, for example, "*it takes at least 30 minutes to download my books*". Moreover, the *Portability* and *Reliability* class both contain all the five types of *behavioral* NFRs.

VI. Discussion

A. Analysis of study results

RQ1: What is the distribution of NFR classes in App user reviews?

Users report Reliability and Usability most frequently in App user reviews. In this study through manually classifying App user reviews, we found that *Reliability* and *Usability* are mentioned most often by users and they together account for around 80% of all user reviews on quality. User reviews on *Reliability* are mainly about the abnormal reaction of the system users observed when they used the App, and as for *Usability*, users mainly mention aspects of user interface and problems about operation. This result indicates that users usually provide quality information about the systems that is visible to them in user reviews. In other words, users often report aspects that directly affect them when using the App, which further validates the fact that *Reliability* and *Usability* are two external quality characteristics that users mainly concern [26]. Users care about whether the system is easy to use, not about whether the system is easy to modify [27]. In contrast, as a typical internal quality characteristic, *Maintainability* is often considered by developers in industrial requirements specifications [12], which can also explain the situation that in the 1278 user review sentences of this study there are no reviews on *Maintainability*.

RQ2: How many NFRs describe system behavior?

Most NFRs in App user reviews describe behavioral properties of the systems. From the results of this study, we can see that over 70% NFRs in App user reviews are classified as *behavioral* NFRs because they describe *behavioral* properties of the systems. Moreover, most of *behavioral* NFRs (around 77.3% of all *behavioral* NFRs) describe *interface* behavior of the systems. Since FRs are known to describe behavior over the interface of the systems [10][12], we can conclude that most NFRs in this study that describe the same type of behavior as FRs do, i.e., *interface* NFRs, are essentially not non-functional. This result has a practical significance. Many studies show that NFRs have not received much attention and software developers usually pay more attention to FRs during software development [7][28]. For researchers and practitioners, this result means that most NFRs can be handled like FRs in requirements elicitation, specification, and analysis, which is also confirmed in [12].

RQ3: Which system view do behavioral NFRs address?

Few NFRs in App user reviews address the system view *architecture*. Our results show that in the 1278 NFRs only 28 NFRs (around 2.2%) describe *architecture* behavior and in the *Usability* class no NFRs are classified as *architecture*. Moreover, most NFRs classified as *architecture* are distributed in the *Reliability* class. We analyzed the 28 *architecture* NFRs in detail and found that they are either time-related or abnormal reactions of the systems observed by users when the interactions among sub-systems or system components happen. It is understandable that although NFRs classified as *architecture* in App user reviews are related to sub-systems or system components, they do not essentially mention the architecture of the App due to the limitation of users' knowledge and experience, consequently it may be difficult for developers to directly obtain information about system architecture from user reviews.

RQ4: In which type of behavior theory are behavioral NFRs expressed?

Users focus on the time-related aspects when they report the *Performance efficiency* characteristic of the

systems. In this study most NFRs (around 51.5% of all *behavioral* NFRs) in App user reviews are classified as *logical* with respect to the *behavior theory* they use, and in the *Security*, *Portability*, *Usability*, and *Reliability* class *logical* NFRs account for the largest part. However, most NFRs in the *Performance efficiency* class are classified as *timed*. We checked the NFRs about *Performance efficiency* in detail and found that users mainly concentrate on the response and processing time of the systems when they report the *Performance efficiency* characteristic. Aside from NFRs that describe the time behavior of the systems, there are also NFRs about the resource utilization and the capacity of the systems which only account for a small portion. This result indicates that users pay more attention to their time-related experience when they use Apps.

B. Comparison of NFRs in App user reviews and industrial requirements specifications

1) Quality characteristics identified in App user reviews and industrial requirements specifications: First of all, we conducted a comparison of the NFR classes between the results of this study and the work in [12] with respect to quality characteristics (ISO/IEC 25010 was used in this study and ISO/IEC 9126 was used in [12]). The distributions of NFR classes in App user reviews and industrial requirements specifications are shown in TABLE IV. We observe that different from the classification results of NFRs in [12], no NFRs in this study are classified as *Functional suitability* or *Maintainability* class, and users primarily concern the aspects of the systems on *Reliability* and *Usability*. Since industrial requirements specifications are formal and professional requirements documents, they usually record more comprehensive aspects of the systems than App user reviews, which also explains that NFRs in industrial requirements specifications mention more quality characteristics than those in App user reviews. In addition, App user reviews include the discriptions about NFRs that have been implemented and NFRs that have not been implemented yet from the perspective of users, while industrial requirements specfications include NFRs to be implemented that reflect the concerns of all stakeholders. This factor is an important reason for the difference on the distributions of varous NFRs from the two sources. For example, we can see that App user reviews care about external quality characteristics such as *Reliability* and *Usability*. While industrial requirements specifications consider most on *Functional suitability* and internal quality characteristics such as *Maintainability*. We should know that users' satisfaction is vital for the success of a software product and it is users who decide whether to use a product [29]. Therefore, developers are supposed to understand the concerns of users to compete in the market [30].

2) The distribution of NFRs classified based on the system view in App user reviews and industrial requirements specifications: TABLE V shows that the distributions of *representational* NFRs and *behavioral* NFRs with respect to the *system view* in App user reviews and industrial requirements specifications. We can see that overall NFRs from the two sources have similar distributions based on the *system view*, and they both have a highest proportion of *interface*. In industrial requirements specifications, NFRs that address *architecture* behavior (around 16.0% of all NFRs) are more than those address *state* behavior (around 7.2% of all NFRs), while in App user reviews there are more NFRs classified as *state* (around 13.8% of all NFRs) than *architecture* (around 2.2% of all NFRs). This result indicates that industrial requirements specifications focus more on the architecture of the systems while users report more state-related aspects of the systems (e.g., the breakdown of the Apps).

TABLE IV. DISTRIBUTION OF NFR CLASSES IN APP USER REVIEWS AND INDUSTRIAL REQUIREMENTS SPECIFICATIONS

Quality characteristic defined in ISO/IEC 9126 (ISO/IEC 25010)	**App User Reviews**	**Industrial Requirements Specifications**
Reliability	45.9%	16.2%
Usability	33.7%	12.8%
Efficiency (Performance efficiency)	9.5%	10.9%
Portability	9.3%	8.7%
Security	1.6%	19.6%

TABLE V. DISTRIBUTIONS OF REPRESENTATIONAL NFRS AND BEHAVIORAL NFRS WITH RESPECT TO THE SYSTEM VIEW IN APP USER REVIEWS AND INDUSTRIAL REQUIREMENTS

		App User Reviews		**Industrial Requirements Specifications**	
	System view	count	%	count	%
Behavioral NFRs	Interface (Black-box)	695	54.4%	273	51.5%
	Architecture (Glass-box)	28	2.2%	85	16.0%
	State (glass-box)	176	13.8%	38	7.2%
Representational NFRs	-	379	29.7%	134	25.3%
Total	-	1278	100.0%	530	100.0%

The overall classification results of NFRs in App user reviews and industrial requirements specifications are shown in Fig. 4. Through comparing the distributions of NFRs with respect to the *system view* in each quality characteristic class, we found that one thing in common in App user reviews and industrial requirements specifications is that no NFRs in the *Usability* class are classified as *architecture* and no NFRs in the *Efficiency* (*Performance efficiency*) class are classified as *state*. We can also see the difference on the distributions of NFRs with respect to the *system view* in certain quality characteristic classes. For example, the distributions of *behavioral* NFRs and *representational* NFRs in the *Security* and *Usability* class between the two sources differ greatly. In the *Security* class, most NFRs in industrial requirements specifications are classified as *representational* (the authors of [12] explained that many *Security* NFRs contain a reference to a standard), while a large part of NFRs in App user reviews are classified as *interface* (most are about the access to their personal information). The *Usability* class in App user reviews has a highest share of NFRs on *representational* (most are about the user interface of the system), while NFRs classified as *interface* account for the largest part of the *Usability* class in industrial requirements specifications. From the perspective of users, they care about whether the App is easy to use and user interface is one of the most direct factors that affect their experiences when using an App. This is a potential reason that users focus on user interface when they report on *Usability*.

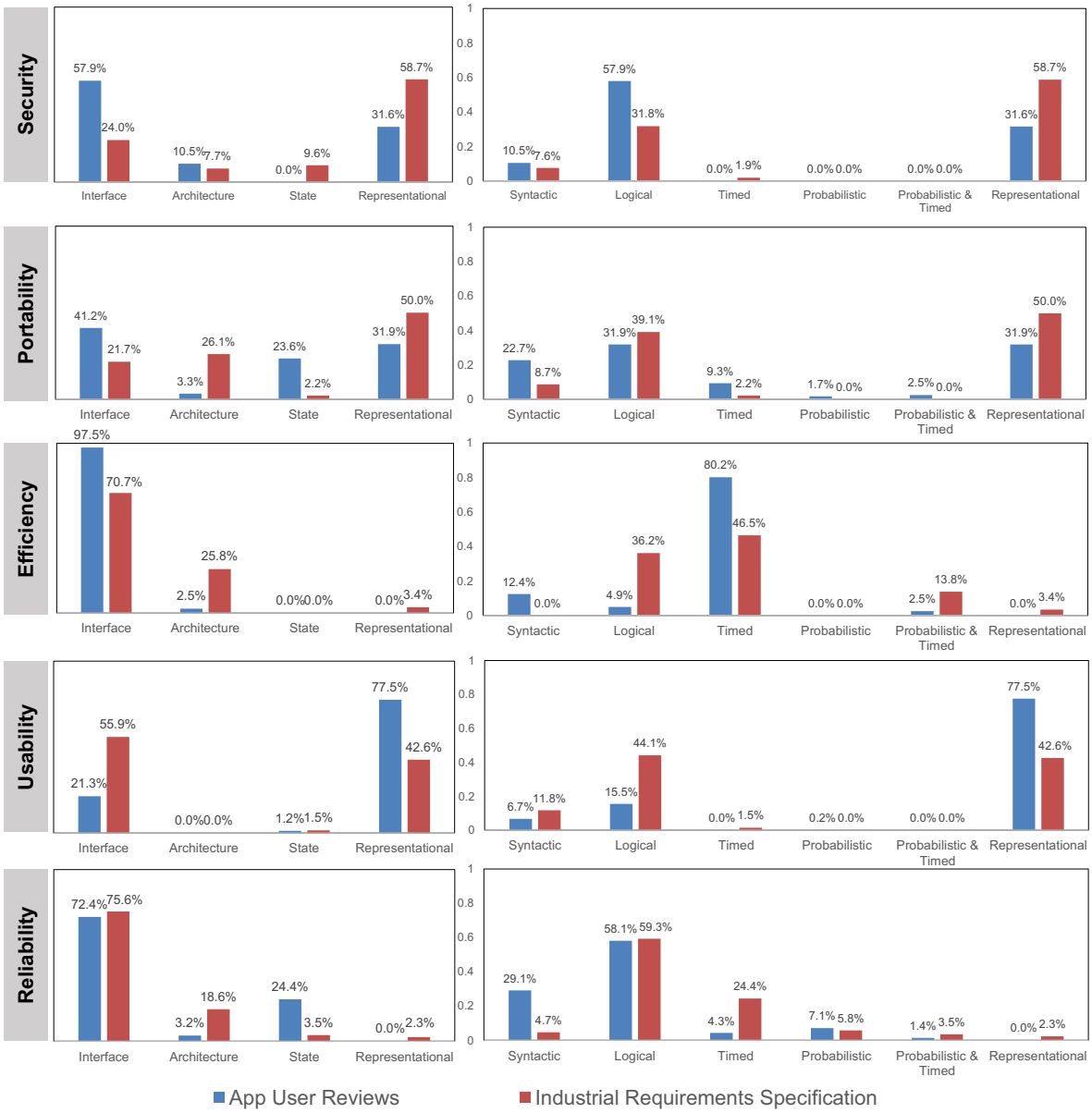

Fig. 4. Distributions of NFRs within the NFR classes based on the system view and the behavior theory in App user reviews and industrial requirements specifications.

3) The distribution of NFRs classified based on the behavior theory in App user reviews and industrial requirements specifications: TABLE VI presents the distribution of NFRs in App user reviews and industrial requirements specifications with respect to the *behavior theory*. The results show that most *behavioral* NFRs from both sources are classified as *logical*. We can also see that the proportion of NFRs classified as *probabilistic* (around 3.5% of all NFRs) or *probabilistic & timed* (around 1.1% of all NFRs) in App user reviews is slightly higher than that of NFRs classified as *probabilistic* (around 1.3% of all NFRs) or *probabilistic & timed* (around 2.1% of all NFRs) in industrial requirements specifications. It is worth mentioning that during our classification process, we found that users used some adverbs of frequency (e.g., "*often*", "*sometimes*") to describe certain situations in reviews. This kind of NFRs are considered as *probabilistic* in our classification. While in industrial requirements specifications, *probabilistic* NFRs seldom use these adverbs and instead they use concrete numbers or proportions to quantify their probability.

As shown in Fig. 4, we found that almost all quality characteristic classes in App user reviews and industrial requirements specifications have a highest share of *behavioral* NFRs on *logical*. The only exception is that the *Efficiency* class in industrial requirements specifications and the *Performance efficiency* class in App user reviews both have a highest share of *behavioral* NFRs on *timed*. This means that the time behavior plays an important role in the *Performance efficiency* characteristic of the systems and both App user reviews and industrial requirements specifications pay more attention to the time-related experience when using

a software product. Moreover, no NFRs of the *Efficiency* class in the industrial requirements specifications are classified as *syntactic*, which accounts for about 12.4% of the *behavioral* NFRs in the *Performance efficiency* class in App user reviews. We checked the NFRs classified as *syntactic* in the *Performance efficiency* class and found that these NFRs describe the resource (e.g., battery power, storage) consumption of Apps, which may not be well-documented in industrial requirements specifications.

TABLE VI. DISTRIBUTIONS OF REPRESENTATIONAL NFRS AND BEHAVIORAL NFRS WITH RESPECT TO THE BEHAVIOR THEORY IN APP USER REVIEWS AND INDUSTRIAL REQUIREMENTS

		App User Reviews		**Industrial Requirements Specifications**	
	Behavior theory	count	%	count	%
Behavioral NFRs	Syntactic	244	19.1%	47	8.9%
	Logical	463	36.2%	277	52.3%
	Timed	133	10.4%	54	10.2%
	Probabilistic	45	3.5%	7	1.3%
	Probabilistic & timed	14	1.1%	11	2.1%
Representational NFRs	-	379	29.7%	134	25.3%
Total	-	1278	100.0%	530	100.0%

VII. THREATS TO VALIDITY

We discuss the threats to validity in this study according to the guidelines in [25] and the measures taken to alleviate the threats.

Construct validity focuses on whether the theoretical constructs are interpreted and measured correctly. In this study, a threat to construct validity is the subjectivity when deciding whether a user review sentence falls into a specific type during the classification process. To alleviate this threat, we used the definitions of NFR types in ISO/IEC 25010 [15] as one of the classification schemes, and we provided clear descriptions of different NFR types in the other two classification themes to help researchers understand their definitions. Moreover, the classification process of user reviews was accomplished by three researchers. A pilot classification of 40 NFRs from App user reviews was conducted by the three researchers independently before the formal classification, and any disagreements were discussed and resolved by the three researchers in order to achieve a consensus among researchers on the classification results. During the classification process, the remaining NFRs were manually classified by the first author and the third author independently. Any disagreements on the classification results were discussed and resolved with the second author. Another threat to construct validity is the way user reviews are written. User reviews usually vary greatly in the structure and style because of the personal knowledge of the users, and some reviews were written ambiguously. In order to alleviate this threat, the dataset has excluded the sentences that are too ambiguous to understand. The third threat to construct validity is whether the NFRs in App user reviews used in this work are persuasive in drawing reasonable conclusions. In order to alleviate this threat, we used a random sample of collected user reviews from two Apps.

Internal validity focuses on the study design, and particularly whether the results follow from the data. One threat to internal validity is that the quality models used to classify NFRs in this study (defined in ISO/IEC 25010) and in [12] (defined in ISO/IEC 9126) are slightly different, which may have an impact on the comparison results and conclusions. However, it is worth noting that ISO/IEC 25010 is an updated version of ISO/IEC 9126. Compared with ISO/IEC 9126, ISO/IEC 25010 adds two quality characteristics: *Compatibility* and *Security* (*Security* is a sub-characteristic of *Functionality* in ISO/IEC 9126), and renames *Functionality* and *Efficiency* to *Functional suitability* and *Performance efficiency* respectively. This difference has a limited impact on the comparison results of this study because there is no *Compatibility* NFRs in the 1278 NFRs from App user reviews (App users seldom mention the relationship between the App they are talking about and other Apps or systems in their reviews). Another threat to internal validity is that the application domains of NFRs in the two sources are different, which may have an impact on the comparison results of this study. Moreover, the information about the development processes of the industrial projects and Apps in this work is not available to us, which may also affect the comparison results and therefore put limitations on the internal validity of this study.

External validity refers to the degree to which the findings of the study can be generalized to other participant populations or settings. In this study, we conducted classifications on the user reviews of two popular Apps: iBooks (iOS) in the books category and WhatsApp (Android) in the communication category. The diversity of the App categories and the platforms can alleviate this threat. However, we have to admit that the number of App categories in this study is limited and it is still unclear whether the classification results in this study are reproducible when using other categories of Apps (e.g., Facebook). Another threat to external validity is that in the results of this study, there are no NFRs classified as certain quality characteristic classes (e.g., *Compatibility*). In the next step, we plan to conduct a more comprehensive study through collecting user reviews from the major categories of Apps in order to alleviate this threat.

Reliability focuses on whether the study yields the same results if other researchers replicate it. In this study, the classification process was manually conducted by three researchers, and a threat to reliability is the personal bias on the classification results. In order to alleviate this threat, we make explicit the classification process and take measures to avoid the personal bias as much as possible (see Section IV-B-2)).

VIII. CONCLUSIONS AND FUTURE WORK

In this study, we classified 1278 NFRs which come from 4000 randomly collected App user review sentences based on the *system view* and the *behavior theory*. In addition, we compared our classification results with those of NFRs in industrial requirements specifications [12]. The goal of this study is to gain a deeper understanding on the nature of NFRs in App user reviews and to further compare the difference on the distributions of various NFRs between App user reviews and industrial requirements specifications. Our classification results show that users report most frequently on *Reliability* and *Usability*. Most NFRs in App user reviews (around 54.4% of all NFRs) describe *interface* behavior of the systems and thus are essentially not non-functional. Moreover, few *behavioral* NFRs (around 2.2% of *behavioral*

NFRs) address the system view *architecture*. Also, through comparison on the distributions of NFRs in App user reviews and industrial requirements specifications, we found that overall NFRs from the two sources have a similar distribution with respect to the *system view* and the *behavior theory*, but the distributions of NFRs classified as *architecture* and *state*, and the distributions of NFRs in certain quality characteristic classes show some differences. For example, the *Usability* class in App user reviews has a highest share of NFRs on *representational* while NFRs classified as *black-box* account for the largest part of the *Usability* class in industrial requirements specifications. The difference on the distributions of the NFRs between App user reviews and industrial requirements specifications shows that the concerns between the two sources on software products are different in certain aspects.

App user reviews is one important source to derive user requirements. In the next step, we plan to carry out our work in the following aspects: (1) The dataset we used in this study comes from two Apps. To improve the generalizability of our work, we plan to increase the number of the categories of the Apps to get more convincing results. (2) In this work NFRs are classified into several quality characteristics defined in ISO/IEC 25010 [15], but they are not further classified into sub-characteristics. In the next step we plan to conduct a finer-grained classification on App user reviews in order to better understand the nature of NFRs in App user reviews. (3) App user reviews reflect the concerns of the users while industrial requirements specifications describe the concerns of all stakeholders, which is one important reason for the difference on the distribution of various NFRs between App user reviews and industrial requirements specifications. We plan to investigate the impact of different focuses of stakeholders on requirements elicitation. (4) Additionally, we want to investigate how various NFRs collected from user reviews can be used to enrich industrial requirements specifications as a complementary source in requirements engineering.

REFERENCES

[1] M. Hosseini, K. T. Phalp, J. Taylor, and R. Ali. 2014. Towards crowdsourcing for requirements engineering. In Proceedings of the 20th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), Empirical Track.

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

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

[4] D. Pagano and B. Brügge. 2013. User involvement in software evolution practice: a case study. In Proceedings of the 35th International Conference on Software Engineering (ICSE), IEEE, 953-962.

[5] L. Chung and J. C. S. do Prado Leite. 2009. On Non-Functional Requirements in Software Engineering. In Conceptual Modeling: Foundations and Applications. Springer, 363-379.

[6] I. Sommerville and P. Sawyer. 1997. Requirements Engineering: A Good Practice Guide. John Wiley & Sons, Inc.

[7] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. 2012. Non-functional Requirements in Software Engineering (Vol. 5). Springer.

[8] D. Mairiza, D. Zowghi, and N. Nurmuliani. 2010. An investigation into the notion of non-functional requirements. In Proceedings of the 25th ACM Symposium on Applied Computing (SAC), ACM, 311-317.

[9] N. Yusop, D. Zowghi, and D. Lowe. 2007. The impacts of non-functional requirements in web system projects. International Journal of Value Chain Management, 2(1): 18-32.

[10] M. Broy. 2015. Rethinking nonfunctional software requirements. IEEE Computer, 48(5): 96-99.

[11] M. Broy. 2016. Rethinking nonfunctional software requirements: A novel approach categorizing system and software requirements. Software Technology: 10 Years of Innovation in IEEE Computer. John Wiley & Sons.

[12] J. Eckhardt, A. Vogelsang, and D. M. Fernández. 2016. Are" Non-functional" Requirements really Non-functional? An Investigation of Non-functional Requirements in Practice. In Proceedings of the 38th International Conference on Software Engineering (ICSE), IEEE, 832-842.

[13] E. C. Groen, S. Kopczyńska, M. P. Hauer, T. D. Krafft, and J. Doerr. 2017. Users - The hidden software product quality experts?: A study on how app users report quality aspects in online reviews. In Proceedings of the 25th International Conference on Requirements Engineering (RE), IEEE, 80-89.

[14] M. Lu and P. Liang. 2017. Automatic classification of non-functional requirements from augmented app user reviews. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE), ACM, 344-353.

[15] ISO/IEC. ISO/IEC 25010, 2011. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models.

[16] E. C. Groen, J. Doerr, and S. Adam. 2015. Towards crowd-based requirements engineering a research preview. In Proceedings of the 21st International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), Springer LNCS, 247-253.

[17] E. C. Groen, N. Seyff, R. Ali, F. Dalpiaz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini, and M. Stade. 2017. The crowd in requirements engineering: The landscape and challenges. IEEE Software, 34(2): 44-52.

[18] D. Pagano and W. Maalej. 2013. User Feedback in the AppStore: An Empirical Study. In Proceedings of the 21st International Conference on Requirements Engineering (RE), IEEE, 125-134.

[19] W. Maalej and H. Nabil. 2015. Bug report, feature request, or simply praise? on automatically classifying App reviews. In Proceedings of the 23rd International Conference on Requirements Engineering (RE), IEEE, 116-125.

[20] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan. 2016. Analyzing and automatically labelling the types of user issues that are raised in mobile App reviews. Empirical Software Engineering, 21(3): 1067-1106.

[21] B. W. Boehm, J. R. Brown, and M. Lipow. 1976. Quantitative evaluation of software quality. In Proceedings of the 2nd International Conference on Software Engineering (ICSE), IEEE, 592-605.

[22] G. C. Roman. 1985. A Taxonomy of Current Issues in Requirements Engineering. IEEE Computer, 1(4): 14-23.

[23] M. Glinz. 2005. Rethinking the notion of non-functional requirements. In Proceedings of the 3rd World Congress for Software Quality (WCSQ), 55-64.

[24] F. L. Li, J. Horkoff, J. Mylopoulos, R. S. Guizzardi, G. Guizzardi, A. Borgida, and L. Liu. 2014. Non-functional requirements as qualities, with a spice of ontology. In Proceedings of the 22nd International Conference on Requirements Engineering (RE), IEEE, 293-302.

[25] F. Shull, J. Singer, and D. I. Sjøberg. 2008. Guide to Advanced Empirical Software Engineering. Springer.

[26] ISO/IEC. ISO/IEC 9126-1, 2001. Software engineering - Product quality - Part 1: Quality model.

[27] S. McConnell. 2004. Code Complete. Pearson Education.

[28] Q. L. Nguyen. 2009. Non-functional requirements analysis modeling for software product lines. In Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering (MiSE), IEEE, 56-61.

[29] A. B. AL-Badareen, M. H. Selamat, M. A. Jabar, J. Din, S. Turaev, and S. Malaysia. 2011. Users' perspective of software quality. In Proceedings of the 10th WSEAS International Conference on Software Engineering, Parallel And Distributed Systems (SEPADS), 84-89.

[30] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan. 2015. What do mobile app users complain about?. IEEE Software, 32(3): 70-77.