Automated Quality Assessment for Crowdsourced Test Reports of Mobile Applications

Xin Chen*†, He Jiang*†, Xiaochen Li*†, Tieke He‡, Zhenyu Chen‡
*School of Software, Dalian University of Technology, Dalian, China
†Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian, China
‡School of Software, Nanjing University, Nanjing, China
chenxin4391@mail.dlut.edu.cn, jianghe@dlut.edu.cn, li1989@mail.dlut.edu.cn
dg1232002@smail.nju.edu.cn, zychen@nju.edu.cn

***Abstract*—In crowdsourced mobile application testing, crowd workers help developers perform testing and submit test reports for unexpected behaviors. These submitted test reports usually provide critical information for developers to understand and reproduce the bugs. However, due to the poor performance of workers and the inconvenience of editing on mobile devices, the quality of test reports may vary sharply. At times developers have to spend a significant portion of their available resources to handle the low-quality test reports, thus heavily decreasing their efficiency. In this paper, to help developers predict whether a test report should be selected for inspection within limited resources, we propose a new framework named TERQAF to automatically model the quality of test reports. TERQAF defines a series of quantifiable indicators to measure the desirable properties of test reports and aggregates the numerical values of all indicators to determine the quality of test reports by using step transformation functions. Experiments conducted over five crowdsourced test report datasets of mobile applications show that TERQAF can correctly predict the quality of test reports with accuracy of up to 88.06% and outperform baselines by up to 23.06%. Meanwhile, the experimental results also demonstrate that the four categories of measurable indicators have positive impacts on TERQAF in evaluating the quality of test reports.**

***Index Terms*—crowdsourced testing, test reports, test report quality, quality indicators, natural language processing**

I. Introduction

Mobile devices grow dramatically and mobile applications evolve rapidly, posing great challenges to the software test activities. However, due to the typical characterizes of mobile devices, such as limited bandwidth, unreliable networks, and diverse operation systems, traditional testing (e.g., laboratory testing) for desktop applications and web applications may be not intrinsically appropriate to a mobile environment [1]. Recently, many companies or organizations tend to crowdsource their software testing tasks for mobile applications to an undefined, geographically dispersed large group of online individuals (namely crowd workers) in a open call form [2], [3]. Therefore, crowdsourced testing has received wide attention from both academia and industry [4]–[7]. In contrast to traditional testing, crowdsourced testing can be performed anytime and anywhere [8], thus tremendously improving the testing productivity. Meanwhile, crowdsouced testing recruits not only professional testers, but also end users for testing [3]. Developers can gain real feedback information, functional requirements, and user experiences.

In crowdsourced testing, crowd workers from open platforms help developers perform testing and submit test reports for abnormal phenomena [4]. A typical test report usually provides some critical field information, such as *environment*, *input*, *description*, and *screenshot* for developers to understand and fix the bug. One of the most important characteristics is that crowdsourced testing is strictly limited in time, such as several days or one week [4]. Thousands of test reports are sent to developers in a short time and the quantity heavily exceeds the available resources to inspect them. Meanwhile, due to the poor performance of workers and the inconvenience of editing on mobile devices, test reports may differ sharply with respect to their quality, which seriously affects the understandability and reproducibility for developers to fix the bugs.

Many studies focus on shortening the total inspection cost by reducing the quantity of inspected test reports [4], [5], [8], [9]. However, these studies neglect the impact of the quality of test reports on the inspection efficiency. High-quality test reports provide overall information and the contained contents can be easily understood, developers can reproduce and fix the bugs within a reasonable amount of time. In contrast, low-quality test reports often lack of important details and consume developers much time and efforts, thus heavily decreasing their efficiency. It is perfect if the quality of test reports can be reliably measured by automated methods so as to developers select the high-quality test reports for inspection. Although no study has been conducted to investigate how to automatically measure the quality of test reports, some studies around quality assessment for bug reports and requirement specifications have thrown light on a practicable direction by defining a set of indicators to quantify the desirable features or properties of bug reports and requirement specifications [10]–[13].

In this paper, to help developers predict whether a test report can be selected to inspect within limited resources, we attempt to resolve the problem of test report quality assessment by classifying test reports as either "Good" or "Bad". We propose a new framework named TEst Report Quality Assessment Framework (TERQAF) to automatically model the quality of test reports. First, Natural Language Processing (NLP) techniques are applied to preprocess test reports. Then, we define a series of quantifiable indicators to measure the desirable properties of test reports and determine the corresponding

978-1-5386-4969-5/18/$31.00 © 2018 IEEE

value of each indictor according to the textual content of each test report. Finally, we transform the numeric value of a single indicator into the nominal value (namely Good, Bad) by means of a step transformation function and aggregate the nominal values of all indicators to predict the quality of test reports.

To evaluate the effectiveness of TERQAF, we perform five crowdsourced test tasks for real industrial mobile applications and collect five datasets with 936 test reports from crowd workers. Developers have spent about one week to inspect and evaluate these test reports. With the help of developers, we form the ground truth for experiments. We employ the commonly used accuracy as the metric and investigate three research questions to evaluate the effectiveness of TERQAF in test report quality assessment. Experimental results show that TERQAF can achieve 88.06% of accuracy in predicting the quality of test reports and outperform baselines by up to 23.06%. Meanwhile, the experimental results also demonstrate that the four categories of measurable indicators have positive impacts on TERQAF in test report quality assessment.

In this study, we make the following contributions:

1) To the best of our knowledge, this is the first work to investigate the quality of test reports and resolve the problem of test report quality assessment.
2) To automatically model the quality of test reports, we propose a new framework named TERQAF by using a taxonomy of quantifiable indicators to measure the desirable properties of test reports.
3) We evaluate TERQAF over five real industrial crowdsourced test report datasets of mobile applications. Experimental results show that TERQAF can accurately predict the quality of test reports.

The rest of this paper is structured as follows. Section II details the background and the motivation. In Section III, we systematically summarize some desirable properties that an expected test report should meet. Section IV defines a taxonomy of indicators for the desirable properties. In Section V, we detail TERQAF for test report quality assessment. The experimental setup and the experimental results are presented in Section VI and Section VII, respectively. Section VIII discusses the threats to validity and Section IX reviews some related work. Finally, we conclude this study in Section X.

II. Background and Motivation

In this section, we introduce the background of crowdsouced testing in detail and present several examples as the motivation for resolving the problem of test report quality assessment.

In crowdsouced testing, companies or organizations are responsible for preparing software under test and testing tasks for crowdsouced testing. Workers passing an evaluation select test tasks according to their mobile devices, perform testing, and edit test reports for the observed abnormal behaviors [4], [5]. These test reports are written in natural language together with some screenshots based on the predefined format. A typical test report is usually composed of different fields, such as *environment*, *input*, *description*, and *screenshot*, some of which may vary slightly in different projects from different crowdsourced platforms, but are generally similar in the content [8], [9]. In our experiments, we perform five crowdsourced test tasks for mobile applications with our industrial partners on the Kikbug crowdsourced testing platform[1].

Table I arrays several examples of crowdsouced test reports from the real industrial data. Notably, in our experiments, all test reports are written in Chinese. In order to facilitate understanding, we translate them into English. Field *environment* is the basic configurations of used mobile devices, including phone type, operation system, screen resolution, and system language. Field *input* lists the concrete test steps which are well designed by workers in performing testing based on the actual test requirements. Developers precisely follow these test steps to reproduce the bugs possibly. Field *description* contains the detailed descriptions of bugs and occasionally involves real user experience. By reading the descriptions, developers can understand the content and make an initial decision for fixing the bugs. Field *screenshot* sometimes provides some necessary images to capture the system symptoms when the bugs occur.

However, for crowdsourced mobile application testing, test reports are generally short and uninformative. For example, TR_1 in Table I only contains two words which may make developers confused to understand the bug. Meanwhile, workers do not strictly comply with the given format to write test reports. They may describe their work details or reveal system bugs in Field *input*. For example, the *input* of TR_2 provides the bug description rather than concrete test steps, thus seriously hampering developers to reproduce the bug. At times, for saving time or other motivations, workers may report multiple bugs in the same test report which is called a multi-bug test report. Generally, multi-bug test reports carry more natural language information but relatively marginal for each contained bug. Also, the test steps may be not sufficiently exact to reproduce each bug. For example, TR_3 is a multi-bug test report which reveals two distinct software bugs. Lines 1 to 3 detail that the system does not work well to remind users how to open the downloaded pictures. Line 4 briefs a sharing problem using only two words. Meanwhile, the test steps are not clearly distinguished to reproduce the two bugs.

In aggregate, test report inspection and evaluation are a significant part of mobile application maintenance. However, the widely varied quality of test reports obviously influences the efficiency of developers. In particular, low-quality test reports usually need more time and efforts to understand, thus some test reports are dealt with extremely slowly or not at all constrained by the limited available resources. In practice, test reports usually contain many duplicates. When facing multiple test reports revealing the same bug, developers should select the high-quality one for inspection. In this paper, to help developers predict whether a test report should be selected to inspect, we attempt to resolve the problem of test report quality assessment. Inspired by existing studies around quality assessment for bug reports and requirement specifications, we

[1] http://kikbug.net