Understanding Android Crowdsourced Worker through Portraits

Yongming Yao[*], *Tongtong Bai, Jiangtao Lu, Changyou Zheng, Song Huang*

Command and Control Engineering College, Peoples Liberation Army Engineering University, China
yaoyongming@aeu.edu.cn, btt070619@163.com, lujiangtao2024@163.com,
zhengcy@aeu.edu.cn, huangsong@aeu.edu.cn

Abstract

Millions of crowdsourced workers are using crowdsourced platforms for their second jobs. They have different testing skills, styles, preferences, etc. Understanding them is important for making collaborative decisions such as crowdsourced task assignments. Existing crowdsourced platforms do not provide enough information about crowdsourced workers, and we need to spend a lot of effort searching for this information on crowdsourced platforms. In contrast to the basic worker information displayed on crowdsourced platforms, we propose describing workers as a quick way to characterize and understand them. We discuss how to build portraits of workers that are concise and informative. We propose a multidimensional model for Android crowdsourced worker portraits to specify attributes about various aspects of software testing. Then, we propose a methodology that utilizes text analytics, web data analytics, and test script analytics techniques to analyze various sources of data about workers on a crowdsourced testing platform in order to construct the portraits. The constructed portraits can be vividly displayed on the web to help people quickly learn about crowdsourced workers and make better decisions when collaborating on testing software. Results show the potential for recommended improvements and correct assignments when using our portraits. Worker portraits are an effective form of characterizing workers. It helps one to quickly understand workers and can be applied to a variety of applications in the software testing process.

Keywords: Crowdsourced testing, Android, Portraits, Worker selection

1 Introduction

In a crowdsourced task, the behavior and attitude of the crowdsourcer are important characteristics that attract people. Crowd workers must perform the task to the best of their abilities since the engagement of crowd workers is significantly dependent on obtaining valuable results. Crowdsourced platforms will recruit suitable individuals from a large pool, and crowd-worker engagement may be influenced by both the crowd-workers' characteristics and the characteristics of the crowd-testing jobs.

To reduce the cost of software testing, improve software quality [1-2], and speed up testing progress, many methods and approaches have been proposed to use crowdsourced software testing to replace or assist in-house testing [3-6]. One of the most fundamental functions is identifying the appropriate worker for a specific testing task [7-9]. It is important to keep in mind that even though crowdsourced resources are cost-effective, they are not entirely free. Therefore, to ensure the sustainable development of crowdsourced technology, it is crucial to allocate tasks efficiently. However, previous technologies have been lacking in terms of considering the continuity between time and historical data, which has resulted in a reduction in the efficiency of task allocation. When it comes to scaling up crowdsourced testing, it is essential to maximize the information available to each member.

Additionally, not all group workers are equally skilled at finding bugs. The wrong employee can miss bugs or report duplicate bugs, and hiring them requires an extraordinary budget. Due to the extensive, uncertain nature of crowdsourced workers, it is difficult to predict outcomes. [10], we should only allow some workers to participate in crowdsourced testing tasks. Therefore, it is valuable to recommend a suitable group of workers for testing tasks so that more software bugs can be detected with fewer workers. To help identify suitable workers for crowd-testing tasks, many different methods have been proposed, e.g., based on the testing environment [9-10], experience [11-13], abilities [9, 14], or expertise on the task [15-20], and so on. Unfortunately, these methods have limited applicability to the volatile crowdsourced software testing process.

Crowd worker characteristics include personnel expectations, extrinsic motivation, and satisfaction motivation. Personnel expectations represent crowd workers' expectations of participating in the crowd-testing task during the event. Deviations from these expectations can affect crowd workers' motivation, commitment, and satisfaction. Before participating in a crowdsourced task, contractors must provide relevant information, including task requirements, completion time, and accurate compensation details for participants. Workers' lack of task information may lead them to take on tasks that do not match their abilities, resulting in wasted time and underpayment. This, in turn, affects crowd workers' motivation and engagement, ultimately reducing overall crowd-worker participation. Crowd workers are extrinsically motivated to participate in crowd-testing

*Corresponding Author: Yongming Yao; E-mail: yaoyongming@aeu.edu.cn

DOI: https://doi.org/10.70003/160792642025052603011

tasks to receive monetary or material rewards. There is a linear relationship between crowd workers' engagement and extrinsic motivation. Another characteristic of crowd workers is satisfaction motivation; they want to improve their skills in participating in crowdsourced tasks or completing them to gain a sense of satisfaction and achievement. Crowd workers rely on improved skills to perform crowdsourced tasks more effectively.

Characteristics of crowd workers' work include task characteristics, crowd worker visibility, and work environment. Characteristics of crowd-worker jobs include task characteristics, crowd-worker visibility, and work environment. Differences in crowdsourced tasks may affect employee engagement, especially if irresponsible crowdsourcers violate the rules of crowdsourced tasks. Crowd workers' performance will improve with feedback related to their tasks, and they will be diligent in performing various tasks. The visibility of crowd workers is the second characteristic; all crowd workers are visible on the crowdsourced platform, and appropriate crowd workers can be identified in a large group of people. The third characteristic of the crowd worker's work characteristics is the work environment, where the work environment and environmental characteristics play a vital role in the crowd worker's execution of tasks. Crowdsourcing platforms provide multiple engagement options for crowdsourcing. Crowd workers can participate virtually from their homes, shops, or workplaces. As potential work environments increase, so do employees' availability, flexibility, and independence, which can also improve performance and satisfaction. As workers and test tasks accumulate on the crowdsourcing platform, a large amount of data is retained, including personal characteristics of the crowdsourced workers, such as name, age, job, and other important information. It also includes characteristic behavioral data such as the number of projects the crowdsourced workers have participated in, the completion of the projects, and the test reports submitted. It is a challenge to improve the match between crowdsourced workers and projects and ensure the dedication of crowdsourced workers through these characteristic data.

In summary, this paper makes the following contributions: we utilize crowd workers' personal information and behavioral characteristic data on a crowd-testing platform. We use user profiling technology to extract user feature labels, establish personal and group portrait models of the crowd workers, and verify the model's effectiveness through simulation experiments. We also compare the results with those obtained from the random selection method to evaluate the accuracy of this new method.

The remainder of this paper is organized as follows. In Section 2, we propose an Android crowdsourced software worker portrait model. Experimental validation of our method is performed in Section 3. Section 4 discusses related work. Section 5 discusses validity threats. Finally, in Section VI, we conclude this work.

2 Android Crowdsourced Software Worker Portrait Model

To construct the user portrait of Android crowdsourced software workers, we limited the type of testing involved to Android testing so that all the relevant behaviors of the crowdsourced software workers are related to Android testing. Specifically, they include the number of times they have participated in executing Android tests, the test scores they have achieved in Android testing, the number of Android defects they have found, the frequency with which they have participated in Android testing, and so on. The situation of the match between crowdsourced software workers' work and the work on Android has a significant impact on the construction of the user portrait of Android crowdsourced software workers.

Therefore, the construction of Android crowdsourced software workers' labels will be highly relevant to Android. At the same time, the Android crowdsourced software workers' personnel user portrait model includes essential information and dynamic information about workers, etc., which can be obtained. Figure 1 shows the labeling system of the Android crowdsourced software worker portrait. Android crowdsourced software worker portrait model consists of two types: basic information and dynamic information. The basic information includes identity/occupation and mobile device used labels, and the dynamic information includes behavioral characteristics labels.

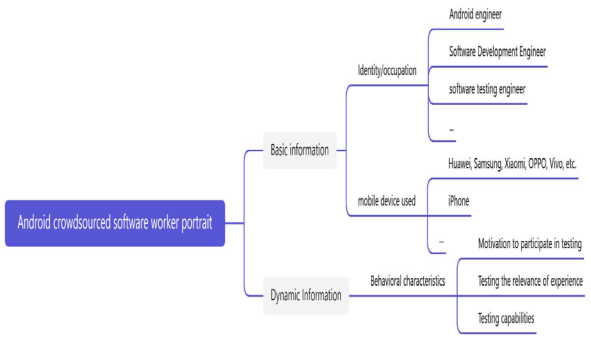

Android crowdsourced software worker portrait
Basic information
Identity/occupation
Android engineer
Software Development Engineer
software testing engineer
...
mobile device used
Huawei, Samsung, Xiaomi, OPPO, Vivo, etc.
iPhone
...
Dynamic Information
Behavioral characteristics
Motivation to participate in testing
Testing the relevance of experience
Testing capabilities

Figure 1. Android crowdsourced software worker portrait labelling system

2.1 Android Crowdsourced Software Worker Label Definition

The user portrait of Android crowdsourced software workers should include two aspects: basic information about the workers, such as identity/occupation and use of devices, and dynamic information extracted from their behavioral data, such as motivation to participate in testing, relevance of their testing experience, and testing ability.

(1) Identity/job is one of the most basic information about workers and a basic reflection of their field. The identity/job characteristic value is agreed to be T_{job}, and it is defined as Equation 1:

$$T_{job} = \begin{cases} L_{skil-j}, R_{Android} \geq L_{skil-j} \\ L_{skil-j-1}, R_{Android} < L_{skil-j} \end{cases} \tag{1}$$

where $R_{Android}$ denotes the worker's work and Android system correlation denotes a degree that the options for Android-related work content are given by the test platform. The selection of compliant options is determined by the worker's work content. The formula for $R_{Android}$ is shown in Equation 2:

$$R_{Android} = \frac{Opt_{Android}}{Opt_{total}} \tag{2}$$

where $Opt_{Android}$ indicates the number of options selected by the worker, and Opt_{total} indicates the total number of options; L_{skil} indicates the skill level embodied in the worker, based on the content of the work selected on the previous working day, and the specific rating is determined by the test platform according to the content of the work of the various levels of Android engineers. By combining the content of the work of the workers and their abilities with each other, it is possible to further categorize the skill level of these people into introductory, beginners, intermediates, and advanced. We can further classify them as beginners, elementary, middle, and advanced. The skill level L_{skil} is expressed as L_{skil-j} (j = 1,2,3,4,5), where $L_{skil-1} = 0.1$, $L_{skil-2} = 0.2$, $L_{skil-3} = 0.4$, $L_{skil-4} = 0.6$, and $L_{skil-5} = 0$.

(2) The role of using the device is extremely important; it is the main source for workers to contact and obtain system information, and it is also the embodiment of workers' familiarity with different operating systems, such as the familiarity with Android, IOS, etc. The characteristic value of using the device, denoted as T_{acq}, is defined as $T_{acq} = L_{use-j}$; L_{use-j} represents the worker's use level of using the Android device. The usage level of the device is classified into novice, average, and expert level according to the different functions of using the Android device. The usage level L_{use-j} is denoted as L_{use-j} (j=1,2,3,4), where $L_{use-1} = 0.1$, $L_{use-2} = 0.3$, $L_{use-3} = 0.5$, and $L_{use-4} = 0.8$.

(3) The extent to which workers are active in Android testing at a crowdsourced software testing platform is captured by the motivation of each worker to participate in testing. The activeness includes the time when the worker starts the first test task, denoted as T_{start}, the time of the last test task completed at this crowdsourced software testing platform, denoted as T_{end}, the number of Android tests that the worker participates in, denoted as $A_{Android}$, and the eigenvalue of the activeness of the participation in Android testing, denoted as T_{act}, This is defined in equation 3, as shown below:

$$T_{act} = \frac{A_{Android}}{T_{end} - T_{start}} \tag{3}$$

(4) The number of tests related to Android testing in the test tasks operated and completed by workers is determined by the relevance of their testing experience. The worker's total number of involved test tasks is denoted as A_{total}, and the eigenvalue of the relevance of testing experience is denoted as T_{rel}, as shown in Equation 4:

$$T_{rel} = \frac{A_{Android}}{A_{total}} \tag{4}$$

(5) One of the most important characteristics of workers is their own testing ability, which directly determines the quality of the test report. Testing ability is reflected in the number of defects found by workers in Android testing and the weight of the defects. The more defects are found, the higher the weight of the defects is, which directly reflects the stronger ability of the workers. Because the importance of each defect is different, the number of defects with different weights is also different; therefore, the average weighted number of defects can be calculated, which is denoted as A_{bug}. Bug_i represents the ith defect found, and according to the different weights of the defects, it can be further classified into the following four levels: no weight, low weight, general weight, high weight. The weight of the defects is denoted as BWL_j(j=1,2,3,4), where $BWL_1 = 0$, $BWL_2 = 0.5$, $BWL_3 = 1.5$, and $BWL_4 = 4.5$. The average weighted number of defects can be calculated as $A_{bug} = \Sigma_{i=1,j=1}^{n,4} Bug_i \times BWL_j / \Sigma_{i=1}^{n} Bug_i$. The eigenvalue T_{abi} of worker's testing ability is defined as $T_{abi} = A_{bug}$.

In the previous section, we described the feature labels in the user portrait model of constructing crowdsourced software workers from the perspectives of some of the workers' attributes, such as occupation/identity, use of equipment, motivation to participate in testing, relevance of testing experience, and testing ability, respectively. When the relevant data of the workers are acquired, the corresponding feature values can be further derived after analyzing the data. However, the weights of these feature values are different. According to the research of An et al. [21] in crowdsourced software testing, the features of crowdsourced software workers satisfy the relationship of "ability > experience > preference", so in the construction of the user portrait model, the importance of the above feature values are ranked as follows: testing ability > testing Relevance of experience > Occupation/identity > Motivation to participate in testing > Use of equipment. According to the different weights of these features, we can quickly determine what type of workers are suitable to participate in Android testing.

It is judged by the worker's competence profile T_{abi}, which combines two key factors that reflect the quality of the test report: the number of defects and the weight of the defects. After calculating the proportion of defects with different weights and their scores, it can be seen that the higher the weight of the defects, the higher the score it accounts for. Therefore, discovering defects with weights and earning scores indicates that the worker is more competent. Based on the value of this feature, the competence level of Android workers can be quickly classified as excellent, medium or mediocre competence.

Judged by the correlation T_{rel} of testing experience, the correlation feature T_{rel} of testing experience combines the total number of tests completed and the number of Android tests completed by the worker. The higher the percentage of Android tests that workers have participated in, the relative ability to respond to the worker's preference and familiarity with testing on Android, and workers with a high degree of familiarity and a tendency to complete Android tests are more likely to get a good test score in Android testing. By evaluating this feature, it is possible to classify workers in Android testing at the experience level into experienced, average experience or lack of experience.

Judging by the workers' own occupation/identity characteristics, if the workers themselves are engaged in some software development work, such as Android engineers, Android software test engineers, etc., they are bound to have a more obvious advantage when they perform this kind of task of Android testing; at the same time, engineers at different levels have their own strengths and weaknesses in their abilities, so the testing platform lets workers choose their usual work content to compare the match with the demand points of the testing tasks. Generally speaking, the closer the workers' work content is to the testing tasks and the more in-depth the skills they have mastered, the better their performance will be when completing this type of testing task. Therefore, the skill levels of workers are categorized as introductory, beginner, intermediate, intermediate-advanced and advanced.

Then, the activity of workers in executing Android testing tasks is judged by the motivation T_{act} of participating in testing, and the total ratio of the number of completed Android tests to the total time of executing all testing tasks reflects the workers' willingness to execute Android testing life and the efficiency of completing the test tasks, and the workers who are willing to execute Android testing and complete the test tasks efficiently will be better in completing the testing tasks. workers will be better at completing the testing tasks. Therefore, through the scores of this characteristic, workers' motivation towards Android testing can be classified into positive, average and negative types.

Finally, through the discussion of the worker's use of the device characteristics T_{act}, different users, the use of Android/IOS device level is actually very different. While most people most commonly used to the phone's communication functions (such as phone calls, emails, text messages), the use of various types of applications (such as chatting, gaming, shopping) and so on, the general public will not be able to use the mobile phone is more biased to the bottom level of the general public will not use the more low-level functions of the mobile phone (such as reinstallation and brushing, modification of applications). Some users will write a custom desktop system, modify the application installation package and other more kernel operations. The use of the device and the ability to understand the test user will also affect the implementation of the test process. Therefore, the workers' use of the device is classified as novice, average, expert, and specialist.

Based on the above order, to gradually delineate the type of workers belonging, the characteristic attributes of Android workers in this test type can be derived, and the user portrait model of workers can be derived.

2.2 Visualization of User Portrait Models for Android Crowdsourced Software Workers

The worker's user portrait model is the result of the feature extraction process of the user portrait, and the final result generated includes the worker's name and his feature combinations. This process is done autonomously by the crowdsourced software testing platform. The purpose of studying the user portraits of Android workers is to get precise information about the worker's capabilities, to make it easier to find a suitable worker to perform the task when performing this type of testing task, and also to make it easier for the worker to be pushed or invited to the testing tasks that are suitable for him. So, to make it easier for the workers and the crowdsourced software testing task senders to perceive the process and use it for their convenience, the workers' previously obtained user portrait model needs to be visualized.

The visualization of the individual worker user portrait model is specific to the worker himself, whose personal user portrait can be seen on his homepage on the crowdsourced software testing platform.

The individual worker user portrait will contain more personalized information, such as an evaluation of the individual's activity in the category of Android testing. These evaluations are based on the amount of time the user spends executing Android testing tasks on the test platform compared to the execution time of workers across the platform, such as an evaluation of the user's activity in the category of Android testing, the number of Android testing tasks the user performs on the test platform, such as an evaluation of the degree of expertise of the worker in the field of Android testing, the achievements of the worker in Android testing tasks, which is an evaluation of his competence in Android testing can be derived. The degree mentioned above of activity, area of expertise, and ability evaluation are used as labels for individual workers to evaluate the behavior of individual workers in Android testing. Some values need to be quantified in numbers, such as average contribution value, average number of defects found, average number of active defects, average revenue earned, etc., in Android testing.

To visualize these contents above into a single worker user portrait, we choose to present the activity level, the area of expertise, and the ability evaluation, which can be evaluated qualitatively. These aspects are presented through a worker graph combining the evaluation of these labels; we can get the labeled graphs of the abbreviated version of the Android worker user portrait, as shown in Figure 2.

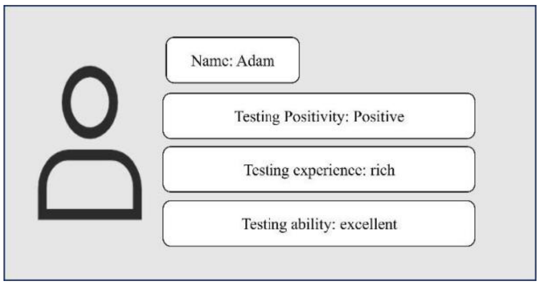


Figure 2. Label for Android worker user portraits

In addition, some other values need to be measured numerically, such as the average contribution value in the field of Android testing, the average number of defects found, the average number of valid defects, the average income earned, and other specific values about the worker's user portrait characteristics that can be displayed in a radar chart. This gives a more intuitive view of the numerical distribution of the values of these characteristics, reflecting the areas in which the performance is outstanding and the areas in which they are lacking. Figure 3 shows an abbreviated version of the Android worker competencies radar chart. The graph expresses the worker's competence in four dimensions, with the darker parts representing the worker's scores.

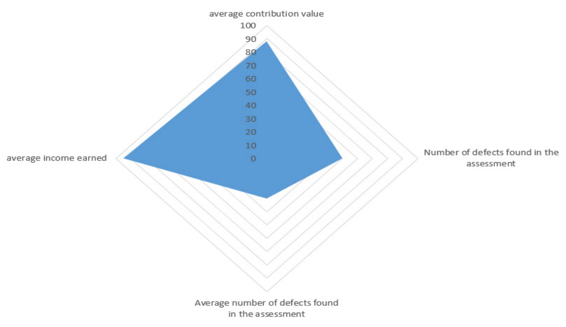


Figure 3. Radar chart of android worker competencies

The group Android worker user portrait is constructed by extracting the user characteristics of all participating workers and refining the user labels for the performance of workers in Android testing, which ultimately serves the testing task contractors and helps them to make decisions.

The visualization of the group Android worker user portrait model is convenient for the test task contractors to browse the workers better. Based on the user portraits given to the workers, contractors can sieve out the workers that meet their current test task. It can be achieved as much as possible with fewer workers, resulting in lower commission to pay for a high-quality test report.

The Android crowdsourced software worker user portrait model constructed in this chapter is for the crowdsourced software testing task contractor. Characteristics such as testing ability, the relevance of testing experience, occupation/status, motivation to participate in testing, and use of devices can be both qualitatively evaluated and quantitatively analyzed. When crowdsourced software testing tasks, it is difficult to quantify the average number of defects found and the average number of practical defects found by workers. Still, using terms such as excellent testing ability and testing experience is more accessible. However, the test task contractor sometimes also needs to see the specific distribution of values, so this design first allows the test task contractor to select labels, such as excellent testing ability, testing experience, and other labels. After clicking on it, a list of workers who meet the requirements will appear in descending order of testing ability. The list will appear in the list of workers in all the task grades related to the labels. When the test task contractor can not be selected, click on the player, which will also appear in the specific value of the distribution of the radar chart. This is the best of both worlds, making it easy for the test contractor to select workers quickly and satisfactorily. Figure 4 shows an abbreviated version of this process.

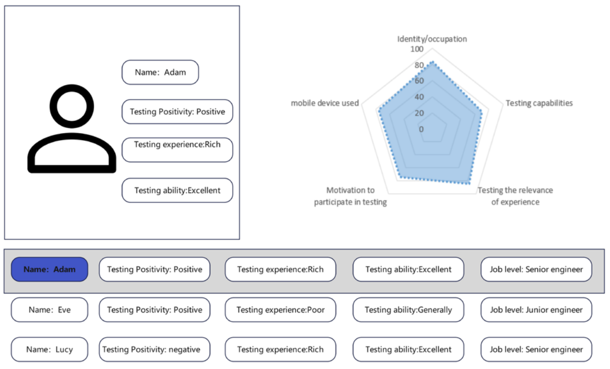

Name: Adam
Testing Positivity: Positive
Testing experience:Rich
Testing ability:Excellent
Identity/occupation
100
80
60
40
20
0
mobile device used
Testing capabilities
Motivation to participate in testing
Testing the relevance of experience
Name: Adam
Testing Positivity: Positive
Testing experience:Rich
Testing ability:Excellent
Job level: Senior engineer
Name: Eve
Testing Positivity: Positive
Testing experience:Poor
Testing ability:Generally
Job level: Junior engineer
Name: Lucy
Testing Positivity: negative
Testing experience:Rich
Testing ability:Excellent
Job level: Senior engineer

Figure 4. A simplified diagram of the selection process for group Android worker competencies

3 Experiment Design and Results Analysis

In this section, we will experimentally validate our approach and design experiments to validate the construction, efficacy, and implementation of user portraits for Android crowdsourced software testing proposed in this paper based on the following two research questions, respectively.

RQ1: Can the method effectively construct portraits of Android crowdsourced software workers?

RQ2: Can Android crowdsourced software worker portraits enhance the effectiveness of crowdsourced software testing projects?

3.1 Experimental Design

In order to answer the above research questions, we design the following two experiments: to validate the effectiveness of the Android crowdsourced software worker portrait model through large-scale analog data simulation, and to verify the enhancement of the user portrait technique on the effectiveness of the crowdsourced software testing project by using the scoring rate, the accuracy rate, and the F-value as the evaluation metrics in comparison with the randomly selecting workers method.

Experimental Design 1: This paper uses the following data and methods to conduct experiments. The current crowdsourced software testing platform has less content about Android testing, and it is difficult to obtain the information and test reports of crowdsourced software workers, so simulated data is used to verify. Firstly, it is assumed that there are 1000 workers on board a certain crowdsourced software testing platform, which has various types of testing tasks, including Android functionality testing, Android compatibility testing, Android security and privacy testing, etc. Secondly, the initial values of the identity/occupational characteristics of the workers, T_{job}, the characteristics of the use of devices, T_{acq}, the eigenvalues of the motivation to participate in testing, T_{act}, and the eigenvalues of the relevance of the trial experience, T_{rel}, etc., are set, and the relevant data are calculated after a certain number of test tasks are simulated.

Experimental Design 2: In order to validate the efficacy of this paper's user portrait technique in an Android crowdsourced software testing project. The effectiveness of this paper's method is compared with the traditional method of randomly selecting crowdsourced software workers. We select five Android application crowdsourced software testing projects from Mooctest crowdsourced software testing platform to conduct experiments. Firstly, n workers are recommended for a crowdsourced software testing project according to the user portrait method in this paper, and in order to ensure the fairness of the experiment, we again use a random selection of the same number (n) of workers. For example, in crowdsourced software testing task 1, the user portrait technique in this paper selects 12 crowdsourced software workers that satisfy the testing requirements based on the user label characteristics, and we again randomly select 12 workers for crowdsourced software testing task 1. Then, for each crowdsourced software worker in each project, we invite the crowdsourced software testing project contractor to score them based on the test results they submit. The ratings were divided into 4 levels, namely very satisfied, satisfied, average, and unsatisfied, and were scored as 3, 2, 1, and 0. Finally, this paper uses the following 3 evaluation metrics for calculation. The formula for calculating the score rate is shown in Equation 5:

$$S = \frac{\Sigma_{i=1}^{n} Value_i}{3 * n} \tag{5}$$

S refers to the score rate, which indicates the degree of satisfaction of the crowdsourced software testing contractor with this testing task. $Value_i$ refers to the score of the ith crowdsourced software worker, i=1−n, where n is the number of people in this crowdsourced software testing task, and $\Sigma_{i=1}^{n}$ $Value_i$ denotes the total score of all selected workers in this task. Since the maximum score of each person is 3, the maximum score of a crowdsourced software testing task is 3*n. For example, in crowdsourced software testing task 1, the scores of 12 randomly selected crowdsourced software workers are 2, 1, 0, 0, 1, 0, 2, 0, 0, 0, 0, 1, and 3. The total score of the crowdsourced software testing task is 10, and the maximum score is 36; the scoring rate is 10/36 = 0.278. The formula for calculating the accuracy is shown in Equation 6:

$$Accuracy = \frac{Number\ of\ people\ satisfied}{total\ number\ of\ people} \tag{6}$$

Accuracy refers to the accuracy rate, which indicates the probability that the recommended crowdsourced software worker meets the requirements of that crowdsourced software testing task. We consider crowdsourced software workers with satisfactory or very satisfactory evaluation results from the crowdsourced software testing task sender to be suitable recommenders (with a score of 2 and above). For example, in crowdsourced software testing task 1, the total number of 12 randomly selected crowdsourced software workers with scores of 2 and above is 3, and the accuracy of their recommendation is 3/12 = 0.25. We consider the above two evaluation metrics to be equally important, so we used the reconciled average of score and recommendation accuracy for our calculations. The F-value is calculated as shown in Equation 7:

$$F = \frac{S * Accuracy * 2}{S + Accuracy} \tag{7}$$

3.2 Analysis of Results

Result Analysis 1: We randomly selected 10 workers out of 1,000 workers to generate a line chart of test data for Android testing, as shown in Figure 5. The horizontal axis represents the worker's characteristic parameters, the vertical axis represents the worker's characteristic value, and the numbers 102, 203, etc. are the numbers of the 10 workers.

Figure 5 shows the testing ability of each worker, which reflects the impact of identity/job T_{job}, equipment usage T_{acq}, enthusiasm for participating in testing T_{act} and test experience correlation T_{rel} and other characteristic values on the final testing ability T_{abi}.

1) Test positivity T_{act}

From the two workers No. 203 and No. 354 in Figure 5(a), there is a big difference between them only in their enthusiasm for participating in testing, and worker No. 354 is higher than No. 203 in terms of test enthusiasm and test experience correlation. Worker No. 203 only has slightly higher identity/work characteristic values than Worker No. 354. As a result, Worker No. 203 performed better in terms of testing ability, indicating that the impact of this characteristic of the enthusiasm for participating in testing T_{act} on testing ability is weak. On identity/job T_{job}.

From the curves of workers No. 835 and No. 994 in Figure 5(b), it can be seen that No. 835 is slightly lower than No. 994 in terms of testing enthusiasm, and there is also some gap between No. 994 and No. 994 in terms of test experience correlation, but No. 994 performed better in the final test. The performance in terms of ability is better, which shows that the influence factor of this characteristic of test participation enthusiasm T_{act} on testing ability is weaker than the test experience correlation T_{rel}.

The characteristic values of workers No. 672 and No. 775 in terms of identity/job T_{job}, device usage T_{acq} and test experience correlation T_{rel} are very close. Worker No. 672 is slightly more enthusiastic about participating in the test than No. 775. Finally, No. 672 staff's testing ability is higher than that of No. 775, which means that when the characteristic values of the other three aspects are close; the enthusiasm for participating in the test T_{act} will affect the performance of the testing ability. If the enthusiasm is high, the performance of the testing ability will be better, so this may be A characteristic value that has a less obvious impact on the testing capability T_{abi}.

2) Influence factors of identity/job T_{job}

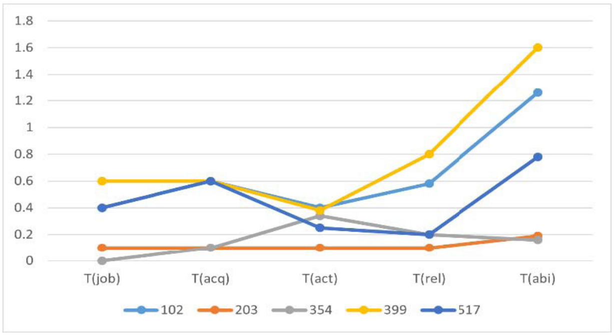
It can be seen from the polyline in the figure that the changes in the two characteristic values of identity/job T_{job} and device usage T_{acq} are positively correlated. Perhaps it is because behind the growth of the eigenvalue of identity/job T_{job} is the improvement of development skills and the improvement of the system. Device understanding, so the impact factor of identity/job T_{job} will be greater than the use of device T_{acq}.

3) Impact factor of T_{rel}

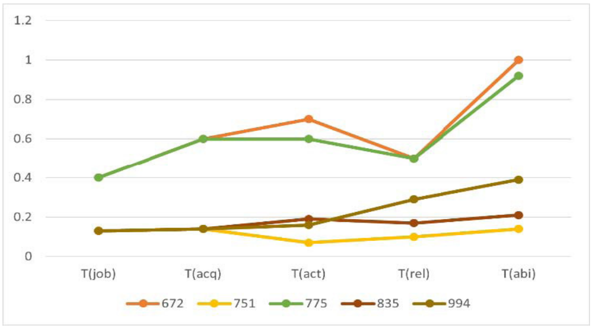
From curves No. 102 and No. 517 in Figure 5(a) and No. 751, No. 835, and No. 994 in Figure 5(b), it can be seen that when the eigenvalues of the identity/job T_{job} and the device T_{acq} are similar, participation the test enthusiasm T_{act} is not much different, and the test enthusiasm T_{act} has a weak impact, so the test experience correlation T_{rel} will significantly affect the final test capability characteristic value, so combined with the previous description, the test experience correlation T_{rel} is the influencing factor The most significant characteristic parameter.

Based on the above data analysis, it can be concluded that it is consistent with the previous hypothesis, that is, the impact weight on Android workers' testing ability is test experience correlation T_{rel} > identity/work T_{job} > testing enthusiasm T_{act} > use device T_{acq}, from which we can effectively build user portraits of Android crowdsourced software workers through defined person labels.

Result Analysis 2: Use our method to compare with the method of randomly selecting workers and use score rate, accuracy rate, and F value as evaluation indicators to verify the improvement of the efficiency of crowdsourced software testing projects by user portrait technology. As seen from Table 1, compared with the random selection method, our method has significantly improved the scoring rate, accuracy rate, and F value. Specifically, in five crowdsourced software testing projects, the scoring rate of this method is maintained at around 60%. Compared with the random selection method, which has an improvement of 12% to 42%, the accuracy rate of this method is maintained at about 70%. compared with the 20%-50% improvement with the random selection method, the F value of this method is maintained at about 65%; compared with the 20%-50% improvement with the random selection method, there is a 16%-46% improvement. To further verify the significant difference between this method and the random selection method, we calculated the p-values of the paired T-test for the three indicators. The results were 0.015, 0.003 and 0.006 respectively. The p-values of the three indicators were all less than 0.05, indicating that our method has significantly improved the energy efficiency of recommending crowdsourced software workers.



(a) No. 102 and other 5 workers' characteristics parameters and eigenvalues



(b) No. 672 and other 5 workers' characteristics parameters and eigenvalues

Figure 5. Performance of 10 workers on Android testing

Table 1. Comparison of random methods and our method

	S		Accuracy		F	
	Random	Our method	Random	Our method	Random	Our method
#1	0.278	0.694	0.25	0.75	0.263	0.721
#2	0.319	0.652	0.304	0.739	0.311	0.693
#3	0.486	0.611	0.458	0.667	0.472	0.638
#4	0.495	0.631	0.459	0.730	0.476	0.677
#5	0.449	0.626	0.388	0.735	0.416	0.676
average value	0.405	0.643	0.372	0.724	0.388	0.681
p-value	0.015		0.003		0.006	

4 Related Work

As a new software testing method, many people in academia and industry have paid continuous attention to crowd testing. The research on crowd testing can be divided into two major categories: one is to study how to improve crowd testing itself and study a series of processes of crowd testing and its related mechanism issues; the other is to study how to use crowd testing as a method Solve different kinds of testing problems or optimize existing testing methods. Research on improving crowd-testing has seen significant development in recent years. Since 2013, research has mainly focused on the integration and decomposition of test reports, the calling and management of workers, and the deconstruction and design of test tasks. However, in crowd testing, there are still some gaps in some areas, such as workers searching for test tasks, test demand parties distributing test tasks, and result verification and reproduction in test reports [20, 22-27].

Yan et al. [27] designed a worker selection algorithm that assigns Web service testing tasks to workers with similar operating environments (including the same network type). They were further improved by the Tempo Spatial Worker Selection algorithm, where the worker's calling time and location are considered in the worker selection activity [28]. Alyahya et al. [29] proposed the Myers-Briggs Type Indicator (MBTI) test to classify different types of workers' personalities. They concluded that extroverted workers are better suited to black box testing techniques, and introverted workers perform white box testing techniques much better than extroverted workers. Leicht et al. [30] studied the impact of task complexity and user expertise on workers' performance in crowd testing. Their results help us understand which testing tasks should be assigned to which group of workers (i.e., experts or novices). They can decide whether a specific type of testing can be crowdsourced or should be conducted using other testing methods. Task complexity is divided into simple tasks and complex tasks. Workers often need to focus on the user interface for simple tasks because simple tasks can identify defects by relying purely on visible input and output. Complex tasks are related to the underlying structure and processes of the software, and workers need to consider how the software should behave when specific steps or operations are performed. Qiang Cui et al. [18] studied the process of selecting workers to participate in crowd testing. They found that workers have an essential role in covering more workers in three aspects: the relevance of test experience, the initiative to participate in testing, and the diversity of test experience. Testing requirements and the detection of more program defects have a significant impact. They designed a way to select workers that combined the above three aspects and verified that the method influenced the Baidu public testing platform. An Gang et al. [21] are building mobile applications. When building a public worker profiling model, we collect workers' historical test data, analyze the workers' behavioral characteristics, test-related experience, and preferences for selecting test types, and conduct experiments based on simulation data to prove that they can improve performance by selecting appropriate workers—test mass accuracy.

5 Threats to Validity

This section analyzes the possible threats to the validity of the conclusions of the user portrait experiment of Android crowdsourced software workers from internal and external validity threats.

Internal validity refers to the extent to which changes in the dependent variable in our experiment are attributable to changes in the independent variable. The design of the user portrait model is mainly affected by the tag weight of crowdsourced software workers. When implementing the method in this article, we refer to the weight priority algorithm proposed by An et al. and reduce this problem through manual analysis and comparison with randomly selecting workers.

External validity refers to how the research results obtained in this experiment can be generalized to other scenarios or environments. The experiments in this article may need to be more generalizable to more general scenarios. To reduce this threat, the data set selected in this article comes from the Mooctest crowdsourced software testing platform, one of China's largest crowdsourced software testing platforms. The selected data set is relatively representative of domestic crowdsourced software testing data.

6 Conclusion

Based on the essential information and behavioral characteristic data of workers, extract user feature tags, build user portrait models for Android crowdsourced software workers, and implement the construction of individual and group user portrait models. Through simulation experiments, verify that the priority of user tag weights meets expectations, indicating that it can effectively construct user portraits; by comparing with the random selection method, our method can improve the efficiency of crowdsourced software testing projects.

References

[1] A. Arya, S. K. Malik, Software Fault Prediction using K-Mean-Based Machine Learning Approach. *International Journal of Performability Engineering*, Vol. 19, No. 2, pp. 133–143, February, 2023.

[2] K. E. Rao, G. A. Rao, P. S. Rao, A weighted ada-boosting approach for software defect prediction using characterized code features associated with software quality, *International Journal of Performability Engineering*, Vol. 18, No. 11, pp. 798–807, November, 2022.

[3] R.-Z. Gao, Y.-B. Wang, Y. Feng, Z.-Y. Chen, W. E. Wong, Successes, challenges, and rethinking–an industrial investigation on crowdsourced mobile application testing, *Empirical Software Engineering*, Vol. 24, No. 2, pp. 537–561, April, 2019.

[4] M. Gasparic, G. C. Murphy, F. Ricci, A context model for IDE-based recommendation systems, *Journal of Systems and Software*, Vol. 128, pp. 200–219, June, 2017.

[5] R. Hao, Y. Feng, J. A. Jones, Y. Li, Z. Chen, CTRAS: Crowdsourced test report aggregation and summarization, *IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, Montreal, QC, Canada, 2019, pp. 900–911.

[6] J. Wang, Y. Yang, R. Krishna, T. Menzies, Q. Wang, iSENSE: Completion-aware crowdtesting management, *IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, Montreal, QC, Canada, 2019, pp. 912–923.

[7] Q. Cui, J. Wang, G. Yang, M. Xie, Q. Wang, M. Li, Who should be selected to perform a task in crowdsourced testing?, *IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Turin, Italy, 2017, pp. 75–84.

[8] J. Wang, S. Wang, J. Chen, T., Menzies, Q. Cui, M. Xie, Q. Wang, Characterizing crowds to better optimize worker recommendation in crowdsourced testing, *IEEE Transactions on Software Engineering*, Vol. 47, No. 6, pp. 1259–1276, June, 2021.

[9] Q. Cui, S. Wang, J. Wang, Y. Hu, Q. Wang, M. Li, Multi-Objective Crowd Worker Selection in Crowdsourced Testing, *The 29th International Conference on Software Engineering and Knowledge Engineering*, Pittsburgh, PA, USA, 2017, pp. 218–223.

[10] I. Oliver, Experiences in the Development and Usage of a Privacy Requirements Framework, *2016 IEEE 24th International Requirements Engineering Conference (RE)*, Beijing, China, 2016, pp. 293–302.

[11] J. Wang, Y. Yang, S. Wang, J. Hu, Q. Wang, Context-and fairness-aware in-process crowdworker recommendation, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 31, No. 3, pp. 1–31, July, 2022.

[12] M. Xie, Q. Wang, G. Yang, M. Li, Cocoon: Crowdsourced testing quality maximization under context coverage constraint, *IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, Toulouse, France, 2017, pp. 316–327.

[13] T. Hossfeld, C. Keimel, C. Timmerer, Crowdsourcing quality-of-experience assessments, *Computer*, Vol. 47, No. 9, pp. 98–102, September, 2014.

[14] Q. Xu, J. Xiong, Q. Huang, Y. Yao, Robust evaluation for quality of experience in crowdsourcing, *Proceedings of the 2013 ACM Multimedia Conference*, Barcelona, Spain, 2013, pp. 43–52.

[15] T. R. A. Giele, T. Mioch, M. A. Neerincx, J.-J. C. Meyer, Dynamic Task Allocation for Human-robot Teams, *International Conference on Agents and Artificial Intelligence*, Lisbon, Portugal, 2015, pp. 117–124.

[16] L. Lyu, M. Kantardzic, T. S. Sethi, Sloppiness mitigation in crowdsourcing: detecting and correcting bias for crowd scoring tasks, *International Journal of Data Science and Analytics*, Vol. 7, No. 3, pp. 179–199, April, 2019.

[17] S. Guo, C. Rong, H. Li, A Real-Time Collaborative Testing Approach for Web Application: Via Multi-tasks Matching, *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Vienna, Austria, 2016, pp. 61–68.

[18] Q. Cui, J. J. Wang, M. Xie, Q. Wang, Towards Crowd Worker Selection for Crowdsourced Testing Task, *Ruan Jian Xue Bao/Journal of Software*, Vol. 29, No. 12, pp. 3648–3664, December, 2018.

[19] S. Alyahya, Crowdsourced software testing: A systematic literature review, *Information & Software Technology*, Vol. 127, Article No. 106363, November, 2020.

[20] H. Jiang, X. Li, Z. Ren, J. Xuan, Z. Jin, Toward Better Summarizing Bug Reports With Crowdsourcing Elicited Attribute, *IEEE Transactions on Reliability*, Vol. 68, No. 1, pp. 2–22, March, 2019.

[21] G. An, T. Zhang, J. Chen, A Mobile Application Crowdsourced Method Based on Behavior Workers Portrait Analysis, *Journal of Northwestern Polytechnical University*, Vol. 35, No. 6, pp. 1083–1088, December, 2017.

[22] N. Wang, L. Cai, M. Chen, C. Zhang, Research Progress in the Processing of Crowdsourced Test Reports, *Testbeds and Research Infrastructures for the Development of Networks and Communications: 14th EAI International Conference*, Changsha, China, 2019, pp. 150–160.

[23] J. Wang, Q. Cui, S. Wang, Q. Wang, Domain adaptation for test report classification in crowdsourced testing, *International Conference on Software Engineering*, Buenos Aires, Argentina, 2017, pp. 83–92.

[24] S. Yu, Crowdsourced Report Generation via Bug Screenshot Understanding, *Automated Software Engineering*, San Diego, CA, USA, 2019, pp. 1277–1279.

[25] J. Wang, S. Wang, Q. Cui, Q. Wang, Local-based active classification of test report to assist crowdsourced testing, *Automated Software Engineering*, Singapore, 2016, pp. 190–201.

[26] Z. Hui, S. Huang, Experience Report: How Do Metamorphic Relations Perform in Geographic Information Systems Testing, *40th IEEE Annual Computer Software and Applications Conference, COMPSAC Workshops* 2016, Atlanta, GA, USA, 2016. pp. 598–599.

[27] M. Yan, H. Sun, X. Liu, ITest: Testing software with mobile crowdsourcing, *the 1st International Workshop on Crowd-based Software Development Methods and Technologies*, Hong Kong, China, 2014, pp 19–24.

[28] M. Yan, H. Sun, X. Liu, Efficient testing of web services with mobile crowdsourcing, *the 7th Asia-Pacific Symposium on Internetware*, Wuhan, China, 2015, pp. 157–165.

[29] S. Alyahya, D. Alrugebh, Process Improvements for Crowdsourced Software Testing, *International Journal of Advanced Computer Science and Applications*, Vol. 8, No. 6, pp. 32–40, 2017.

[30] N. Leicht, M. Rhyn, G. Hansbauer, Can Laymen Outperform Experts? The Effects of User Expertise and Task Design in Crowdsourced Software Testing. *24th European Conference on Information Systems, (ECIS)*, Istanbul, Turkey, 2016, pp. 1–11.

Biographies

Yongming Yao received the B.S. degree in communication engineering from Nanjing University of Posts and Telecommuni-cations in 2010 and the M.S. degree in computer system architect-ture from Xi'an University of Posts and Telecomm-unications in 2013. He received the Ph.D. degree in software engine from Army Engineering University of PLA. He is currently an assistant professor of software engineering with the Software Testing and Evaluation Center, Army Engineering University of PLA. He has contributed more than 10 journal articles to professional journals. His current research interests include software testing, quality assurance, and empirical software engineering. He is a member of CCF.

Tongtong Bai received the B.S. degree in software engineering from Southwest University of Science and Technology in 2018 and the M.S. degree in software engineering from Southwest University of Science and Technology in 2023. He is currently pursuing the Ph.D. degree in software engineering at Army Engineering University of PLA. He previously served as a Performance Test Engineer at NetEase. His research interests are in the area of intelligent software testing and autonomous vehicle testing.

Jiangtao Lu received the B.S. degree in Network Engineering from Xiamen University of Technology in 2016 and the M.S. degree in Computer Technology from the Xiamen University of Technology in 2020. He is currently pursuing the Ph.D. degree in software engineering at Army Engineering University of PLA, with a focus on open-source software vulnerability PoCs.

Changyou Zheng received the Ph. D. degree in millitary information from PLA University of Science and Technology in 2013. He is currently a teacher at Army Engineering University of PLA. His research interests are in the area of software testing, codeblocks.

Song Huang received the Ph.D. degree from the PLA University of Science and Technology. He is currently a Professor of software engineering with the Software Testing and Evaluation Center, Army Engineering University of PLA. He has contributed more than 100 journal articles to professional journals. His current research interests include software testing, quality assurance, data mining, and empirical software engineering. He is a member of CCF and ACM. He is also a member of the advisory boards of Journal of Systems and Software and IEEE Transactions on Reliability.