IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 18, 2021, accepted May 14, 2021, date of publication May 19, 2021, date of current version June 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3081592

A Study on Testers' Learning Curve in Crowdsourced Software Testing

YONGMING YAO[1,2], SONG HUANG[1], CHENG ZONG[1,3], ERHU LIU[1,4], AND NING CHEN[2]

[1]Command and Control Engineering College, People's Liberation Army Engineering University, Nanjing 210007, China
[2]Tongda College, Nanjing University of Posts and Telecommunications, Yangzhou 225127, China
[3]Jiangsu Nuclear Power Corporation, Lianyungang 222042, China
[4]Chinese People's Liberation Army, 94973

Corresponding author: Song Huang (hs0317@163.com)

This work was supported by the National Key Research and Development Program of China under Grant 2018YFB1403400.

ABSTRACT Recommending effective testers in crowdsourced software testing is a challenge. In this paper, we study the improvement of crowdsourced software testers' skills over time. We propose the project difficulty coefficient to eliminate the influence of the item on the tester's score. The hyperbolic learning curve model and exponential learning curve model are used to fit the learning ability of the testers. The experimental results show that when the test data is large, the exponential learning curve can better simulate the improvement of testers' skills.

INDEX TERMS Crowdsourced software testing, learning curve, mooctest.

I. INTRODUCTION

In the process of software product testing, the software product manager hopes to obtain a large amount of feedback as soon as possible, to repair the software product defects as quickly as possible, and improve the software product quality [1]. However, the huge cost of recruiting testers often makes it difficult to recruit large numbers of professional testers. In addition, due to the overlapping of software products' speed, especially the tight life cycle of application products, the cycle time of software testing is sharply compressed. Therefore, how to get test feedback quickly, especially feedback from a large number of real users to help improve the product, and at the same time finish the test task with low cost and high efficiency, is one of the difficulties in the current software test field. Crowdsourced can effectively solve this difficult problem in the field of software testing over a very large range [2]. Crowdsourced is a distributed problem-solving and production organization model brought about by the Internet. In 2006, Howe [3] first proposed the concept of crowdsourced. ''Crowdsourced'' is a distributed problem- solving mode in which a company or organization outsources the work tasks previously performed by full-time employees to a group of non-specific solution providers in a free and voluntary manner through an open Web platform. There are a vast number of people on the Internet, and they can participate in projects as workers [4]–[6]. Crowdsourced software testing [7] refers to taking everyone as a worker and using a large number of people or communities on the Internet to solve the testing problem so that people with different equipment and testing environment can test the same app. The crowdsourced model takes the characteristics of Internet crowd wisdom as the starting point, standardizes the integration of resources in various industries, makes testing more efficient, and solves the problem of understaffed and idle enterprises, thus providing a new method for software testing. Compared with our previous test method for the measurement of a significant change is the task of the test to the Internet, the main reason is that they are different types of test tasks or different with different attitudes and skills. The learning efficiency of the test will be different because of personal reasons.

According to the different initiative, relevance, and diversity of mass testing workers, Cui *et al.* [8] proposed a selection method that considers the three aspects of work at the same time to select appropriate mass testing personnel for each test task, thus improving the defect detection rate and critical point coverage of testing requirements, reducing the testing cost, and improving the testing efficiency.

Crowdsourced software testing is an emerging testing method that has drawn extensive attention from both the industrial and academic communities. With the rise of many successful crowdsource software testing platforms (such as BaiduMTC, uTest, Mooctest, etc.), testing activities have

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaojun Li.

become more efficient. Workers can select the tasks they are interested in to execute and submit test reports and describe the behaviors and discovered defects of the software system under test. Through this kind of form, all the workers help the company to develop and test measurement engineers that find the defects of the software system. Compared with traditional testing activities, the significant change brought by crowd testing is to entrust the tasks of crowd testing to the people testing workers, who can be located in different regions and have different experience and skills in software testing.

Although there are many candidates in crowdsourced software testing, it is not possible to allow all of them to perform test tasks due to cost constraints. Therefore, who performs a test task in crowdsourced software testing, the detection of defects, and the coverage of key test requirements are very important. Crowdsource software test projects to people on the network for execution, and ask testers to submit corresponding test defects. However, the number of testers required for each test project has different requirements on the capabilities of testers. Participating testers have different abilities, and the repeated or irregular test defects are also submitted. The crowdsourced test platform will reward the testers according to their performance. The testers will stay on the platform for testing in order to reward or improve their testing ability.

How will the test level of the tester be measured, and is there any regularity in the improvement of the tester's learning ability? The research of this paper focuses on the study of the learning ability of crowdsourced software testers. A learning curve model is proposed to fit the testing skills of testers. In this paper, our main contributions are as follows:

1. We demonstrate that the testing ability of crowdsourced testers in the Mooctest platform will be improved after participating in multiple testing tasks. As far as we know, we are the first to consider this in crowdsourced software testing.
2. We introduce the learning curve model to the tester testing level of crowdsourced software testing and use the hyperbolic learning model and exponential learning model to model the testers' scores.
3. By modeling the two learning curves, we find that the hyperbolic learning model can better fit the testing level of testers.

The remainder of this paper is organized as follows. In Section II, we propose a background knowledge of crowdsourced software testing, Mooctest, which is a crowdsourced software testing platform, and the learning curve model. Section III presents our study of using learning curves to describe skill improvement. Section IV discusses the related work. Finally, in Section VI, we conclude this work.

II. BACKGROUND

A. CROWDSOURCED SOFTWARE TESTING

In the field of crowdsourced software engineering, researchers have proposed a large number of relevant application technologies and application fields. Recently, Mao *et al.* [4] provided a comprehensive overview of the field of crowdsourced software engineering, covering a large number of literature on crowdsourced software engineering and the introduction of relevant platforms. Crowdsourced software engineering utilizes a large amount of potential, pending online work of people in the form of public summon, bearing the behavior of external software engineering tasks. Zhang *et al.* [9] also put forward crowdsourced software testing. Crowdsourced software testing is an emerging technology. In the last decade or so, many software testing activities have adopted crowdsourcing, such as usability testing [10], [11], functional testing [5], [12], and performance testing [13], [14].

Crowdsourced software testing is required in many situations. For example, sometimes the company's test staff cannot detect all the vulnerabilities of the system, and then crowdsourced security testing can use the white hat to detect vulnerabilities. In addition, usability testing by end-users can reveal more problems. For example, a teacher using educational management software may provide more realistic usability feedback than a software tester with no teaching management experience. In addition, tester pools can help run compatibility tests.

As shown in Figure 1, the test task is published by the task publisher to the test platform, and the crowd worker selects the test task from the test platform and downloads it to his/her own device to complete the test. After the test is completed, the crowd worker submits the test report, which is finally fed back to the task publisher via the test platform.

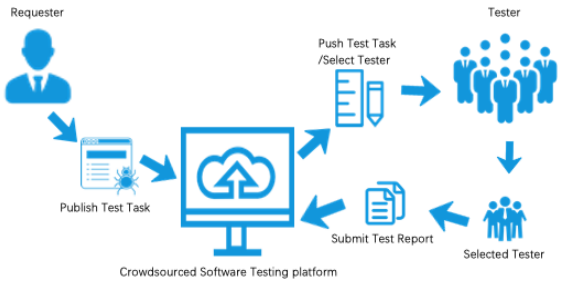

FIGURE 1. The procedure of crowdsourced software testing.

B. MOOCTEST

Mooctest mainly provides mobile application testing and Web application testing services. For four consecutive years, the platform has been used as a platform for national college student software testing competitions, attracting more than 390 universities and 41,000 undergraduate and graduate students majoring in computer science. It has become one of the largest crowdsourced software testing platforms in China.

There are three aspects of crowdsourced software testing projects in Mooctest. One is developer-centered, source-code oriented to Java, C/C++, Python, and other mainstream programming languages; the tester should write corresponding

TABLE 1. A detailed description of the dataset on the mooctest platform.

User_ID	Project name	Submission Order	Submission time	The score for the test script	Participating test points	Total number of test points	Number of bugs	Score of bug
Jam_001	Agile Collaboration Management System Version 1.0	1	2019-01-18 19:35:13	0	0	10	0	0
Jam_001	Agile Collaboration Management System Version 1.0	2	2019-01-18 19:40:41	27	2	10	1	10
Jam_002	Agile Collaboration Management System Version 1.0	1	2019-01-18 19:50:25	45	5	10	2	20
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

test scripts according to the given requirements and jointly evaluate the test effectiveness with coverage rate and bug detection rate. Another kind of test project is Android or IOS application projects; the tester should design test cases given test requirements, write Appium test scripts, find bugs, and complete defect report writing, so as to jointly evaluate test effectiveness with requirement coverage rate and bug detection rate [15]. The third is testing is Web application projects; the tester should design test cases for given test requirements, write Selenium test scripts and Jmeter test scripts, discover bugs, and complete defect report writing, so as to jointly evaluate the test effectiveness with the rationality of the test scheme, requirement coverage rate, and bug detection rate.

We have collected 68 test projects on the Mooctest platform for the period of June 2018-June 2019, and the number of participants is small due to the small number of developers involved in unit testing projects, mainly mobile application testing projects, and web testing projects. The testers on the Mooctest write test scripts by writing test case code and obtain scores based on the quality of the test scripts written. They also submit bugs accordingly.

The dataset in the table 1 is collected from the Mooctest platform, which contains User_ID, Project name, Total number of test points, Submission Order, Submission time, Score for the test script, Participating test points, Number of bugs, Score of bug. There is a lot of data in MOOCTEST, and we give an example in table 1 to describe the data. A tester whose User_ID is Jam_001, participated in the crowdsourced software test project is Agile Collaboration Management System Version 1.0. There are 10 test points in the project. He clicked the submit button at 2019-01-18 19:35:13. However, his test script did not cover the test steps in the project test requirements document, so the score was 0, and the score for the test points was 0, and he did not submit a bug, so the bug score was also 0.

C. THE LEARNING CURVE [16]

The learning curve is a mathematical description of a worker's performance on a repetitive task. As repetition progresses, workers tend to take less time to perform tasks due to familiarity with operations and tools and the discovery of shortcuts to perform tasks. Dr. Wright discovered the learning curve by observing and summarizing the experience of the aircraft production industry, an important economic theory that is an analysis of labor productivity and efficiency. Some industrial sectors are known for using log-linear regression models and modified models to simulate employee learning processes, namely semiconductor factories, electronics, and aerospace component manufacturers, chemical industries, truck assemblers, and automobile parts manufacturing.

In the LC model, the measurement criteria of worker performance as a dependent variable include the time to produce one unit, the number of units produced per time interval, the cost of producing one unit, and the percentage of nonconforming units. LC parameters can be estimated by a nonlinear optimization program to minimize the sum of squares error. When dealing with nonlinear regression problems, the initial value of parameters may be changed if convergence cannot be achieved. The goodness of fit of the model can be estimated by determining the coefficient, the sum of squares of errors, or the consistency of the model with the validated sample.

Among the univariate models, the log-linear model, the exponential model, and the hyperbolic model are the most important.

D. THE LOG-LINEAR MODEL [17]

Wright's model, also known as the ''log-linear regression model'', was the first formal LC model. The model has the following mathematical representation:

$$y = C_1 x^b \tag{1}$$

where Y is the average time (or cost) required to produce x units, and C_1 is the time (cost) required to produce the first unit. Parameter b ($-1 < b < 0$) is the slope of LC, which is used to describe the learning rate of workers. The value of B close to 1 indicates a high learning rate and rapid adaptation to task execution.

E. THE EXPONENTIAL MODEL [18]

Compared with the log-linear model, the exponential model relies more on a complete set of parameters. Compared with log-linear models, these parameters can extract additional

information about the worker's learning process, resulting in more accurate productivity estimates.

Knecht was credited with pioneering work on the exponential learning curve by synthesizing exponential and log-linear functions to improve the ability to predict long-term production operations. The formula is given below:

$$y = C_1 x^b e^{cx} \tag{2}$$

where C is the second constant, and the other parameters are as defined previously.

Three kinds of exponential LC models are often discussed: three-parameter exponential, two-parameter exponential, and constant time model. The three-parameter exponential LC model is as follows:

$$y = k\left(1 - e^{-\frac{x+p}{r}}\right) \tag{3}$$

where y is used to describing the performance of workers by the number of items produced after x units of operating time (y⩾0 and x⩾0). Three parameters are set in the formula. LC: k in Equation (3), which is the maximum performance of workers at the end of the learning process, represented by the number of items produced in each operating time (k⩾0). P corresponds to the previous experience of the worker assessed in units of time (P ⩾0);.Learning rate R is also given units of time.

F. THE HYPERBOLIC MODEL

Mazur and Hastie [19] proposed an LC model for the number of eligible units versus the total number of units produced. In this model, x describes the number of qualified units, and r is the number of unqualified units. Thus, y corresponds to the fraction of the eligible elements times the constant k. The formula is given below:

$$y = k\left(\frac{x}{x+r}\right) \tag{4}$$

The three-parameter hyperbolic model formula is given below:

$$y = k\left(\frac{x+p}{x+p+r}\right) \tag{5}$$

where y is the number of projects generated by x units of the operation time, k is the highest performance level, and R is the learning rate. Mazur and Hastie also proposed adding the parameter P to Equation (4) to allow for the worker's previous experience in performing the task.

III. THE LEARNING CURVE MODELS OF TESTERS'PERFROMANCE

A. PROJECT DIFFICULTY COEFFICIENT

The difficulty of projects released by crowdsourced test platforms is different. Some projects are test versions, which need to test more function points and have more bugs. Naturally, there are more bugs found in a limited time [20]–[22]. However, some projects are release versions, and there are fewer function points to be tested. Many bugs have been fixed, and there are fewer defects. Naturally, there are fewer bugs found in a limited time.

People are required differently in different projects, and projects can vary. There are a variety of people working on the Internet. Some testers are good at app testing, some are good at security testing, and some are just functional testing. There are different types of developer testing, mobile application testing, and web application testing on the muting platform. Different projects in each category have different levels of difficulty. Harder project testers naturally submit lower test reports and scores, and simpler project testers submit higher test reports and scores. It is not possible to distinguish how easily the project affects the tester. We combine the number of people participating in the project with the number of people finding test function points to calculate how easy the project is. Use the item difficulty factor [23] to cancel the impact of difficulty on the tester's score.

The ability of testers on a crowdsourced software testing platform is affected by the number of projects they are involved in. The more time and effort a tester puts into a project, the faster his ability will improve. The testers' ability to improve their skills will also be influenced by their previous experience, age, and learning efficiency of personal information such as tests, and it may also be affected by the number of testers, the difficulty of the tests, the limits on test participation, the total number of test tasks, the time (hours) and use of testers' learning, the number of submissions, and the type of test tasks (as well as preferences and experience).

When Wang *et al.* [24] analyzed the ability scores of developers on TopCoder, they found that the improvement of developers' skills was not completely related to time, but also related to other factors such as the difficulty of tasks. We should also exclude the influence of other factors when we study the skill improvement of crowdsourcing software testers.

In order to more clearly reflect the ability of current testers, the influence of external test conditions, such as test difficulty, on each score of testers is excluded. The difficulty of each test task, the total number of test tasks, and the number of test participants are considered comprehensively. Regrade the overall score for each test to avoid the influence of other factors.

The project difficulty coefficient(PDC) is set as η, where $\eta \in [0, 1]$, and $s_i (0 \leq s_i \leq s)$ refers to the total number of people who successfully completed the test point $i (0 < i \leq n)$. Calculate the PDC:

$$\eta = \frac{\sum_{i=1}^{n} \frac{s - s_i}{s}}{n} \tag{6}$$

In formula 6, i is the number of test points covered in this test item, n is the maximum number of test points in the test item, S is the total number of people participating in the test item, and Si is the number of test points measured Number of people.

The number of specific test points can be calculated by submitting the bugs of each test point in the data set.

For example, 50 people participate in a test project, $S = 50$, and the default test points are 10, $n = 10$. Five people submit valid bugs at the first test point, $S_1 = 5$, and 20 people submit valid bugs at the second test point, $S_2 = 20$. By analogy, we can get $S_3 = 10$, $S_4 = 10$, $S_5 = 15$, $S_6 = 20$, $S_7 = 10$, $S_8 = 10$, $S_9 = 20$, $S_{10} = 40$. Then the difficulty coefficient can be calculated according to formula 6, $\eta = 0.68$.

The comprehensive score is calculated by multiplying the PDC by the sum of the bug score and the test script score. In the same PDC, the higher the test script score and bug score of the tester, the higher the comprehensive score. The calculation of the comprehensive score is as follows:

$$E = \eta \times (S_{score} + S_{bug}) \tag{7}$$

where E is the comprehensive score, S_{score} is the test script score, and S_{bug} is the effective bug score.

For some users who still obtain high scores in the difficult test, the results will be more than 100, and the results will be calculated according to the highest score of 100. If there is a decimal point in the result, it will be omitted.

The testers who participated in the same five projects were selected from the Mooctest, and their five scores were calculated according to the above method. Table 2 shows the comprehensive scores of 10 randomly selected testers after calculation.

TABLE 2. The score calculated by PDC.

Person	1st	2nd	3rd	4th	5th
01	14	43	68	92	100
02	54	68	76	81	87
03	0	28	35	58	66
04	3	21	46	74	90
05	38	63	74	79	89
06	60	66	66	80	91
07	23	26	17	33	40
08	10	14	35	28	56
09	51	30	68	71	95
10	24	40	60	79	93

It can be seen from Figure 2 that the scores of the 10 testers increased with the increase in the number of participants. It can be seen that after eliminating the impact of project difficulty, with the increase of the number of times involved in the project, the score of the testers in the project will also be higher.

Spearman correlation coefficient [25] is a nonparametric (distribution independent) rank statistical parameter, which was proposed by Spearman in 1904 to measure the strength of the relationship between two variables.

As can be seen from Table 3, the correlation coefficient between the test project and 1, 2, 3, 4, 5, and 10 was 1.000 with a significant 0.01 level, which showed that there was a significant positive correlation between the test project and them. The correlation coefficient value between the test project and 6 was 0.975 and showed a 0.01 level of

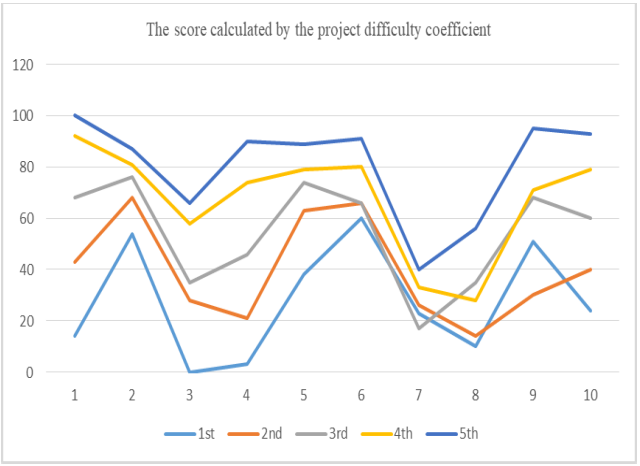

FIGURE 2. The score was calculated by PDC. The X-axis represents the number of testers, and the Y-axis represents the score of the project.

TABLE 3. Correlation relationship between final score and score of difficulty coefficient of project.

Person	Spearman	p-value	Pearson	p-value
1	1	0	0.985	0.002
2	1	0	0.976	0.004
3	1	0	0.98	0.004
4	1	0	0.996	0
5	1	0	0.957	0.011
6	0.975	0.005	0.951	0.013
7	0.7	0.188	0.726	0.165
8	0.9	0.037	0.912	0.031
9	0.9	0.037	0.842	0.073
10	1	0	0.998	0

significance. Therefore, there was a significant positive correlation between the test project and 6. The correlation coefficient between the test project and 7 was 0.700, close to 0, and the p-value was $0.188 > 0.05$, indicating that there was no correlation between the test project and 7. The correlation coefficient between the test project and 8 was 0.900 with a significance level of 0.05, indicating that there was a significant positive correlation between 8 and 8. The correlation coefficient between the test project and 9 was 0.900 with a significance level of 0.05, indicating that there was a significant positive correlation between the test project and 9.

Pearson correlation coefficient [26] is a measure of the degree of linear correlation between two random variables. It was proposed by Karl Pearson in the 1880s.

It can be seen from Table 3 that correlation analysis is used to study the correlation between test items and 10 items including 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10. Pearson correlation coefficient is used to represent the strength of the correlation. Concrete analysis shows the following: the correlation coefficient between the test item and 1 was 0.985 with a significance level of 0.01, indicating that there was a significant

positive correlation between test item and 1. The correlation coefficient between the test item and 2 was 0.976 with a significance level of 0.01, which indicated that there was a significant positive correlation between test item and 2. The correlation coefficient between test items and 3 was 0.980 with a significance level of 0.01, indicating that there was a significant positive correlation between test items and 3. The correlation coefficient value between test item and 4 was 0.996 with a significant 0.01 level, indicating that there was a significant positive correlation between the test item and 4. The correlation coefficient between test items and 5 was 0.957 with a significance level of 0.05, indicating that there was a significant positive correlation between test items and 5. The correlation coefficient between test items and 6 was 0.951 with a significance level of 0.05, indicating that there was a significant positive correlation between test items and 6. The correlation coefficient between test item and 7 was 0.726, close to 0, and the P value was $0.165 > 0.05$, which indicated that there was no correlation between the test item and 7. The correlation coefficient between test items and 8 was 0.912 with a significant 0.05 level, indicating that there was a significant positive correlation between test items and 8. The correlation coefficient between test item and 9 was 0.842, close to 0, and the P value was $0.073 > 0.05$, which indicated that there was no correlation between the test item and 9. The correlation coefficient between test items and 10 was 0.998 with a significance level of 0.01, indicating that there was a significant positive correlation between test items and 10.

B. THE HYPERBOLIC LEARNING CURVE

■ Formula and variable design

For the tester ω, $p_\omega (p_\omega \geq 0)$ is used to represent the tester's current level and experience, and $r_\omega (0 \leq r_\omega \leq 1)$ is the tester's learning efficiency ($q_\omega = \frac{1}{r_\omega}$). According to the hyperbolic learning curve, the formula is modified, and the tester's ability $Q_\omega(x)$ can be defined as:

$$Q_\omega(x) = K \frac{x + p_\omega}{x + p_\omega + q_\omega} \tag{8}$$

As for the factors affecting the testability result of the testers themselves, it is difficult to obtain accurate data related to their specialty and age. Since the projects are the same, this experiment is conducted with the level and learning efficiency of the testers as variables. Set two variables in the formula and the tester's level, previously verified as p_ω, r_ω, to describe the tester's learning speed.

According to Equation (8), the test points submitted by each tester are x, and the number of these submissions is $Q_\omega(x)$. It can be estimated that the tester's learning speed is r_ω and the value of his prior knowledge is p_ω.

■ Fitting model

In order to construct $Q_\omega(x)$, Equation (8) is deformed as follows:

Divide both sides of the equation by K (K>0), and the formula becomes:

$$\frac{Q_\omega(x)}{K} = \frac{x + p_\omega}{x + p_\omega + q_\omega} \tag{9}$$

Take the reciprocal of both sides:

$$\frac{K}{Q_\omega(x)} = 1 + \frac{q_\omega}{x + p_\omega} \tag{10}$$

Subtract 1 from both sides, and the formula becomes:

$$\frac{k}{Q_\omega(x)} - 1 = \frac{q_\omega}{x + p_\omega} \tag{11}$$

Take the reciprocal of both sides again, and the formula becomes:

$$\frac{Q_\omega(x)}{K - Q_\omega(x)} = \frac{1}{q_\omega} x + \frac{p_\omega}{q_\omega} \tag{12}$$

In Equation (12), let $Z_\omega(x) = \alpha_\omega x + \beta_\omega$, $\alpha_\omega = \frac{1}{q_\omega}$, $\beta_\omega = \frac{p_\omega}{q_\omega}$, and $r_\omega = \frac{1}{q_\omega}$. The linear model can be obtained as $Z_\omega(x) = \alpha_\omega x + \beta_\omega$. Using the linear regression method, the $Q_\omega(x)$ scatter points are fitted. According to the formula described in the previous section, the values of α_ω and β_ω can be estimated. The $Q_\omega(x)$ scatter fitting model is shown in Figure 3.

■ Data analysis

According to the least-squares linear regression, estimate the values of α_ω and β_ω, and calculate the p_ω and r_ω. The goodness of fit is a measure of how well the regression line fits the observations. The statistical measure of goodness of fit is the coefficient of determinability (also known as the coefficient of determination) R^2, the maximum value of which is 1. A value of R^2 close to 1 indicates that the regression line is a good fit to the observed values, while a smaller value of R^2 indicates that the regression line is a poor fit to the observed values. The p_ω, r_ω, and R^2 values of the 10 testers are shown in Table 4.

TABLE 4. The values of p_ω, r_ω, and R^2.

Person	p_ω	r_ω	R^2
01	-1.91047	20.842	0.5902
02	-0.35545	1.3175	0.9545
03	-1.25646	0.4874	0.9486
04	-1.7334	2.0518	0.7501
05	-1.00006	1.7015	0.8749
06	-0.97796	1.9281	0.7146
07	1.59293	0.0877	0.5902
08	-1.05924	0.2549	0.7421
09	-1.67982	3.7938	0.5819
10	-1.65473	2.9035	0.7195

According to the exponential learning model to calculate the previous experience of p_ω, r_ω values beyond the range of 0 to 1 have improved, but the error shows that while a hyperbolic linear model has a certain improvement, it still does not accurately simulate the measurement of platform testers' learning ability of ascension.

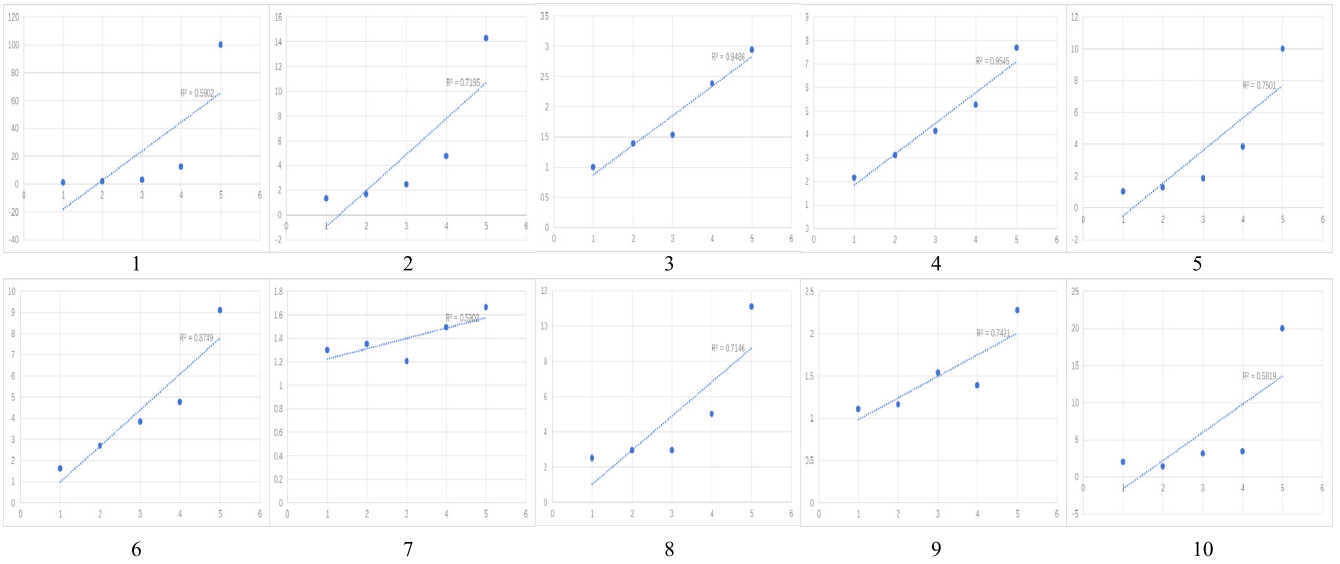

FIGURE 3. The hyperbolic learning model for 10 testers. The x-axis represents the number of tests, and the y axis represents the value of $Z_\omega(x)$.

C. THE EXPONENTIAL LEARNING CURVE

■ Formula and variable design

For the tester ω, $p_\omega(p_\omega \geq 0)$ is used to represent the tester's current level and experience, and $r_\omega(0 \leq r_\omega \leq 1)$ is the tester's learning efficiency ($q_\omega = \frac{1}{r_\omega}$). According to the exponential learning curve, the formula is modified, and the tester's ability $Q_\omega(x)$ can be defined as:

$$Q_\omega(x) = K\left(1 - e^{-\frac{x+p_\omega}{q_\omega}}\right) \tag{13}$$

Set two variables in the formula and the tester's level, previously verified as p_ω, r_ω, to describe the tester's learning speed.

According to Equation (13), the test points submitted by each tester are x, and the number of these submissions is $Q_\omega(x)$. It can be estimated that the tester's learning speed is r_ω, and the value of his prior knowledge is p_ω.

■ Fitting model

In order to construct $Q_\omega(x)$, Equation (13) is deformed as follows:

Divide both sides of the equation by K (K>0), and the formula becomes:

$$\frac{Q_\omega(x)}{K} = 1 - e^{-\frac{x+p_\omega}{q_\omega}} \tag{14}$$

Move the 1 on the right to the left and then multiply both sides by -1, and the formula becomes:

$$1 - \frac{Q_\omega(x)}{K} = e^{-\frac{x+p_\omega}{q_\omega}} \tag{15}$$

Take the log of both sides:

$$\ln\left(1 - \frac{Q_\omega(x)}{K}\right) = -\frac{x+p_\omega}{q_\omega} \tag{16}$$

Finally, the formula is transformed into:

$$-\ln\left(1 - \frac{Q_\omega(x)}{K}\right) = \frac{1}{q_\omega}x + \frac{p_\omega}{q_\omega} \tag{17}$$

In Equation (17), let $Z_\omega(x) = -\ln\left(1 - \frac{Q_\omega(x)}{K}\right)$, $\alpha_\omega = \frac{1}{q_\omega}$, $\beta_\omega = \frac{p_\omega}{q_\omega}$, and $r_\omega = \frac{1}{q_\omega}$. The linear model can be obtained as $Z_\omega(x) = \alpha_\omega x + \beta_\omega$. Using the linear regression method, the $Q_\omega(x)$ scatter points are fitted. According to the formula described in the previous section, the values of α_ω and β_ω can be estimated. The $Q_\omega(x)$ scatter fitting model is shown in Figure 4.

■ Data analysis

According to the least-squares linear regression, estimate the value of α_ω and β_ω, and calculate p_ω and r_ω. The p_ω, r_ω, and R^2 values of the 10 testers are shown in Table 5.

TABLE 5. The values of p_ω, r_ω, and R^2.

Person	p_ω	r_ω	R^2
01	-1.391699537	1.1662	0.8816
02	1.620859298	0.3049	0.9942
03	-0.993325918	0.2697	0.975
04	-1.397277228	0.5656	0.9285
05	0.273291925	0.4025	0.9754
06	1.03614115	0.3514	0.824
07	2.550167224	0.0598	0.5608
08	-0.717215662	0.1609	0.792
09	-0.633859714	0.5446	0.7166
10	-0.96494243	0.5819	0.9227

According to the exponential learning model to calculate the previous experience of p_ω, r_ω values beyond the range of 0 to 1 have improved, but the error shows that while a

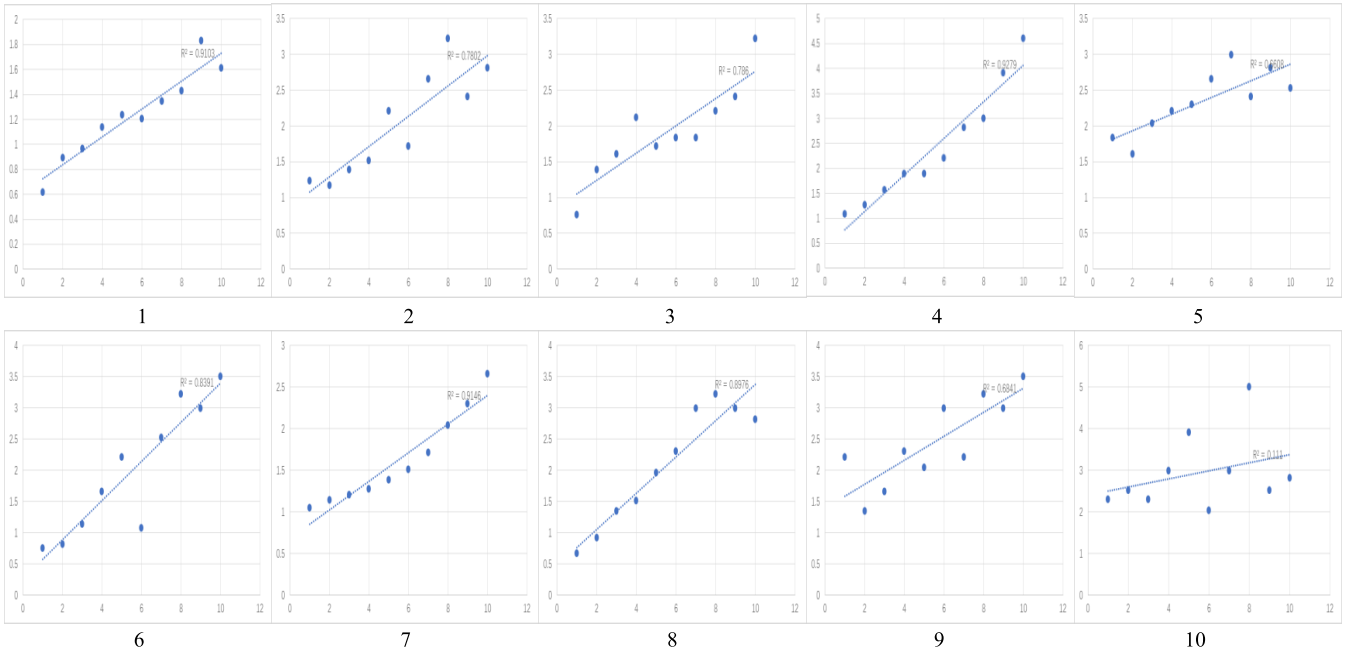

FIGURE 4. The exponential learning model for 10 testers. The x-axis represents the number of tests, and the y axis represents the value of $Z_\omega(x)$.

hyperbolic linear model has a certain improvement, it still does not accurately simulate the measurement of platform testers' learning ability of ascension.

D. COMPARISON

From Figures 3 and 4, we can find that the fitting effect of the exponential learning curve is better than that of the hyperbola learning curve. The discrete points are closer to the fitting curve, which is the exponential learning curve. The 10 sets of data used by the two models are the same. Table 6 shows the comparison of the two models.

TABLE 6. Comparison of the parameter between the two models.

Person	$p_{1\omega}$	$r_{1\omega}$	R_1^2	$p_{2\omega}$	$r_{2\omega}$	R_2^2
01	-1.91047	20.842	0.5902	-1.39169	1.1662	0.8816
02	-0.35545	1.3175	0.9545	1.62085	0.3049	0.9942
03	-1.25646	0.4874	0.9486	-0.99332	0.2697	0.975
04	-1.7334	2.0518	0.7501	-1.39727	0.5656	0.9285
05	-1.00006	1.7015	0.8749	0.27329	0.4025	0.9754
06	-0.97796	1.9281	0.7146	1.03614	0.3514	0.824
07	1.59293	0.0877	0.5902	2.55016	0.0598	0.5608
08	-1.05924	0.2549	0.7421	-0.71721	0.1609	0.792
09	-1.67982	3.7938	0.5819	-0.63385	0.5446	0.7166
10	-1.65473	2.9035	0.7195	-0.96494	0.5819	0.9227

Compared with the hyperbolic model, the negative value of p_ω in the exponential model is reduced, but there is still a negative phenomenon. Based on the actual situation of the Mooctest platform, this phenomenon may be due to the fact that most of the test users are students in school, and most of them have no previous relevant experience. In addition, the number of each person participating in the test is small, leading to a small amount of data in each group of experiments, and there is large uncertainty. Two models in comparison, the exponential model of p_ω and r_ω, are more in line with expectations, and the error is smaller, so the exponential model is more suitable for describing the tester's ability to learn. To verify this, the next section will select the testers whose initial score is higher and verify the simulation of the tester many times.

E. IMPROVED EXPERIMENT

In the case of randomly selected testers, the p_ω value was negative due to low initial performance or low participation in projects. In the selection of testers, we choose experienced testers. Their experience is judged by the test script score and bug score in the dataset. Because experienced testers can skillfully use the test framework and have certain test ideas, their test script score and bug score are high. we selected 10 projects that the testers had good initial scores and participated in together. Table 7 shows 10 testers who have a higher score after calculation by PDC.

Due to the limited space, only parameter comparison tables of the two models are presented here instead of two images with scatter fitting.

As can be seen from Table 8, the p_ω value in the hyperbolic model is still negative, which is not consistent with the actual testers' initial experience. The p_ω value in the exponential model is positive, which is consistent with the actual testers having some initial experience, and the learning rate of r_ω is also consistent with the actual. This experiment once again verified that, compared with the hyperbolic learning model,

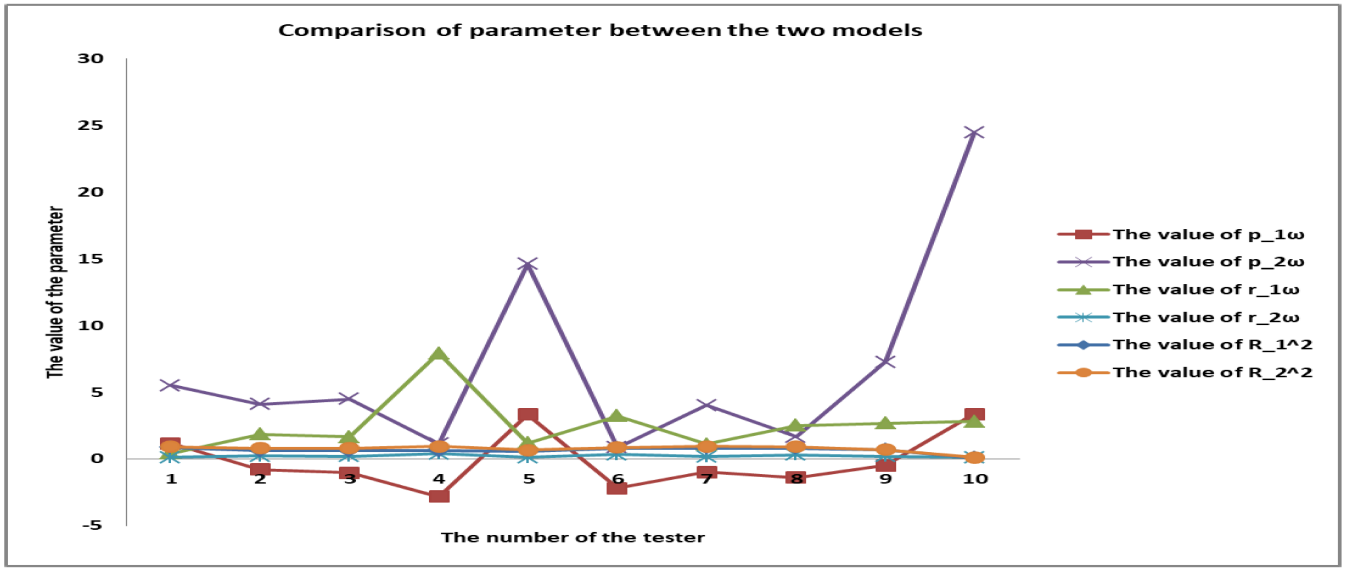

FIGURE 5. Values of two model parameters.

TABLE 7. A higher initial score calculated by PDC.

Person	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
01	46	59	62	68	71	70	74	76	84	80
02	71	69	75	78	89	82	93	96	91	94
03	53	75	80	88	82	84	84	89	91	96
04	66	72	79	85	85	89	94	95	98	99
05	84	80	87	89	90	93	95	91	94	92
06	53	56	68	81	89	66	92	96	95	97
07	65	68	70	72	75	78	82	87	90	93
08	49	60	74	78	86	90	95	96	95	94
09	89	74	81	90	87	95	89	96	95	97
10	90	92	90	95	98	87	95	100	92	94

TABLE 8. Comparison of the parameter between the two models.

Person	$p_{1\omega}$	$r_{1\omega}$	R_1^2	$p_{2\omega}$	$r_{2\omega}$	R_2^2
1	1.15065	0.3923	0.8452	5.497312	0.1116	0.9103
2	-0.79492	1.8476	0.6079	4.099622	0.2118	0.7802
3	-1.06471	1.6396	0.5981	4.475693	0.1913	0.786
4	-2.84457	7.9221	0.6065	1.117117	0.3663	0.9279
5	3.315776	1.1638	0.5597	14.5794	0.1165	0.6608
6	-2.21	3.2114	0.7783	0.835773	0.3142	0.8391
7	-0.99328	1.0865	0.7799	4.006421	0.1713	0.9146
8	-1.40714	2.4839	0.7801	1.637302	0.2906	0.8976
9	-0.49519	2.6638	0.6894	7.244664	0.1921	0.6841
10	3.356751	2.8042	0.0892	24.42523	0.0983	0.1110

the exponential learning curve can better describe the learning ability of testers in the crowdsourced test platform under the condition of large data volume, and it is more suitable to represent the improvement of testers' skills.

As can be seen in Figure 5, hyperbolic learning curve fitting does not work well for previous experience. Seven of the previous experience values are negative, which does not correspond to the p_ω greater than 0. When the exponential learning curve is fitted, all of the previous experience values are greater than 0 and the values are very high, which corresponds to the situation of the tester we selected. For the learning rate, the set learning rate should be between 0 and 1, while nine values in the hyperbolic model are greater than 1. Therefore, it does not conform to the actual situation. The learning rate of the exponential learning curve is between 0 and 1, mostly concentrated around 0.3, so the exponential learning curve model is more suitable for fitting to the testers of the crowdsourced software testing.

IV. RELATED WORK

In crowdsourced software testing, there are many researchers studying the recommendations of testers. Cui *et al.* [27] proposed a multi-target crowd worker selection method (MOOSE) to select crowd testers by maximizing the coverage of testing requirements, minimizing the cost, and maximizing the bug detection experience of the selected group workers. The experiment found that the selected workers could improve the detection rate by 17%. Wang *et al.* [24] proposed a recommended method for the skill improvement of crowdsourced software developers, modeled the skill improvement of developers through the learning curve model, and confirmed that the negative exponential learning curve was suitable for describing the skill improvement of employees. Wang *et al.* [28] proposed a new characterization of crowd workers that leverages testing environment, competencies, and domain knowledge. Based on this, they proposed the multi-targeted crowd worker recommendation method (MOCOM), which aims to recommend a minimum number

of crowd workers for a crowdsourced testing task that can detect the maximum number of bugs.

The learning curve has a wide range of applications and can be used to improve the skills of researchers and testers; it can also play a certain role in prediction and guidance. Bach *et al.* used the learning curve to test the learning performance of the robot. Jinn *et al.* proposed the best business benefit and the best economic benefit according to the model learning curve. Bohn and Terwiesch [29] evaluated the effect of learning throughout the production process of a new product model. Using an improved log-linear LC method, Kannan and Palocsay [30] compared the productivity of different cell layouts in the experiment. In addition, Franceschini and Galleto [31] used the LCs to estimate the reduction in nonconforming product patterns in juice production plants as workers' skills improved. Zong *et al.* [32] proposed a new automatic log analysis method to improve the efficiency of fault analysis based on the knowledge learning ability of fault analysis engineers in the nuclear power industry. The warehouse material order picking system uses artificial intelligence and automation technology, according to the learning curve of artificial language selection and semi-automatic selection, to accelerate the learning process of manual operation automation [33]. There are applications of the learning curve in various fields. Due to the limited space, these are some examples. Our work is to apply the learning curve in crowdsourced software testing to testers in terms of learning skills improvement. To our knowledge, this is also the first time it has been used in crowdsourced software testing workers.

V. THREATS TO VALIDITY

Threats to external validity are relevant to the generality of this study. In the first place, our experimental data come from one of the largest crowdsourced testing platforms in China. Our findings may not be extensible beyond the environment in which we conducted our experiments. However, we used a variety of data to control for this threat.

In the second place, the main threat to conceptual validity consisted of two parameters. These two parameters are designed according to different learning curve models: a priori experience in testing, i.e., testers in the test platform are usually involved in test projects and test project accumulation can improve their skills, so a certain amount of experience is assumed to better match the testing reality. By conducting experiments on both random and high initial skills, it can be found that higher initial skills can be a better fit, but since most of the testers in the crowdsourced test platform are students, the test data is relatively homogeneous, which will have some influence on the experimental results. Also, due to the age, major, and test preference of the testers, the learning time and forgetting curve cannot be collected for the time being. We do not consider the influence of these factors here.

VI. CONCLUSION AND FUTURE WORK

In this paper, we studied how crowdsourced software testers' skills are improved over time. Specifically, we proposed the project difficulty coefficient to eliminate the influence of the item on the tester's score. The hyperbolic learning curve model and exponential learning curve model were used to fit the learning ability of the testers. The experimental results showed that when the test data is large, the exponential learning curve can better simulate the improvement of testers' skills. At the same time, considering the initial experience and learning efficiency of different testers for projects in attended more tests, more test tasks on testers testing skills can better simulate the testers' skills upgrading, in order to more effectively recommend suitable projects for the test platform.

As the data used in this paper is from the Mooctest platform, students in a school account for the majority of the data, and their testing experience is not rich enough, resulting in the negative value of the initial experience fitted. The number of tests that users participate in is not enough, which may bring uncertainty. In future work, we will acquire more data sets to further analyze and simulate the improvement of skills. We will consider other factors such as the learning time of testers. In addition, the forgetting phenomenon of testers will be simulated by using the forgetting curve model, and the conclusion will be more realistic.

REFERENCES

[1] W. E. Wong, X. Li, and P. A. Laplante, "Be more familiar with our enemies and pave the way forward: A review of the roles bugs played in software failures," *J. Syst. Softw.*, vol. 133, pp. 68–94, Nov. 2017.
[2] R. Gao, Y. Wang, Y. Feng, Z. Chen, and W. Eric Wong, "Successes, challenges, and rethinking–an industrial investigation on crowdsourced mobile application testing," *Empirical Softw. Eng.*, vol. 24, no. 2, pp. 537–561, Apr. 2019.
[3] J. Howe, *Wired 14.06: The Rise of Crowdsourcing*. London, U.K.: Wired Magazine, 2006.
[4] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *J. Syst. Softw.*, vol. 126, pp. 57–84, Apr. 2017.
[5] D. Archambault, H. Purchase, and T. Hoáfeld, *Evaluation in the Crowd. Crowdsourcing and Human-Centered Experiments*, vol. 10264. Cham, Switzerland: Springer, 2017.
[6] H. Jiang, X. Li, Z. Ren, J. Xuan, and Z. Jin, "Toward better summarizing bug reports with crowdsourcing elicited attributes," *IEEE Trans. Rel.*, vol. 68, no. 1, pp. 2–22, Mar. 2019.
[7] S. Alyahya, "Crowdsourced software testing: A systematic literature review," *Inf. Softw. Technol.*, vol. 127, Nov. 2020, Art. no. 106363.
[8] Q. Cui, J. J. Wang, M. Xie, and Q. Wang, "Towards crowd worker selection for crowdsourced testing tas," *Ruan Jian Xue Bao/J. Softw.*, vol. 29, no. 12, pp. 3648–3664, 2018.
[9] X. B. Zhang XF, Y. Feng, D. Liu, and C. ZY, "Research progress of crowdsourced software testing," *J. Softw.*, vol. 29, no. 1, pp. 69–88, 2018.
[10] D. Liu, R. G. Bias, M. Lease, and R. Kuipers, "Crowdsourcing for usability testing," *Proc. Amer. Soc. Inf. Sci. Technol.*, vol. 49, no. 1, pp. 1–10, 2012.
[11] Q. Ismail, T. Ahmed, K. Caine, A. Kapadia, and M. Reiter, "To permit or not to permit, that is the usability question: Crowdsourcing mobile Apps' privacy permission settings," *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 4, pp. 119–137, Oct. 2017.
[12] K. T. Chen, C. C. Wu, Y. C. Chang, and C. L. Lei, "A crowdsourceable QoE evaluation framework for multimedia content," in *Proc. ACM Multimedia Conf., Co-Located Workshops Symp.*, 2009, pp. 491–500.
[13] M. Yan, H. Sun, and X. Liu, "ITest: Testing software with mobile crowdsourcing," in *Proc. 1st Int. Workshop Crowd-Based Softw. Develop. Methods Technol.*, 2014, pp. 19–24.
[14] Z. Hui, S. Huang, C. Chua, and T. Y. Chen, "Semiautomated metamorphic testing approach for geographic information systems: An empirical study," *IEEE Trans. Reliab.*, vol. 69, no. 2, pp. 657–673, Oct. 2020.

[15] J. X. Peng Yang Hongyu Sheng, Y. Huang, and J. Xu, "Crowdsourced testing ability for mobile apps: A study on mooctest," *Int. J. Performability Eng.*, vol. 15, no. 11, pp. 2944–2951,2019.
[16] L. M. Surhone, M. T. Tennoe, S. F. Henssonow, H. Ebbinghaus, and T. P. Wright, *Learning Curve*. Betascript, 2010.
[17] H. J. Khamiss, "Log-linear model analysis of the semi-symmetric intraclass contingency table," *Commun. Statist. - Theory Methods*, vol. 12, no. 23, pp. 2723–2752, Jan. 1983.
[18] J. H. Davis, *Exponential Model*. Hoboken, NJ, USA: Wiley, 2011.
[19] J. E. Mazur and R. Hastie, "Learning as accumulation: A reexamination of the learning curve," *Psychol. Bull.*, vol. 85, no. 6, p. 1256, 1978.
[20] N. Kojima, S. Shirato, and Y. Ohno, "Frequency doubling technology threshold visual field results vary with software version," *Jpn. J. Ophthalmol.*, vol. 51, no. 6, pp. 448–452, Nov. 2007.
[21] G. S. Varadan, "Trends in reliability and test strategies," *IEEE Softw.*, vol. 12, no. 3, p. 10, May 1995.
[22] P. K. Chittimalli and M.-J. Harrold, "Recomputing coverage information to assist regression testing," *IEEE Trans. Softw. Eng.*, vol. 35, no. 4, pp. 452–469, Jul. 2009.
[23] H. F. Dingman, "The relation between coefficients of correlation and difficulty factors," *Brit. J. Stat. Psychol.*, vol. 11, no. 1, pp. 13–17, May 1958.
[24] Z. Wang, H. Sun, Y. Fu, and L. Ye, "Recommending crowdsourced software developers in consideration of skill improvement," in *Proc. 32nd IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Oct. 2017, pp. 717–722.
[25] C. Spearman, "The proof and measurement of association between two things. By C. Spearman, 1904," *Am. J. Psychol.*, vol. 100, nos. 3–4, pp. 441–471, 1987.
[26] I. Cohen, Y. Huang, J. Chen, and J. Benesty, "Pearson Correlation Coefficient," Tech. Rep., 2009.
[27] Q. Cui, S. Wang, J. Wang, Y. Hu, Q. Wang, and M. Li, "Multi-objective crowd worker selection in crowdsourced testing," in *Proc. 29th Int. Conf. Softw. Eng. Knowl. Eng.*, Jul. 2017, pp. 218–223.
[28] J. Wang, S. Wang, J. Chen, T. Menzies, Q. Cui, M. Xie, and Q. Wang, "Characterizing crowds to better optimize worker recommendation in crowdsourced testing," *IEEE Trans. Softw. Eng.*, early access, May 23, 2020, doi: 10.1109/TSE.2019.2918520.
[29] R. E. Bohn and C. Terwiesch, "The economics of yield-driven processes," *J. Oper. Manage.*, vol. 18, no. 1, pp. 41–59, Dec. 1999.
[30] V. R. Kannan and S. W. Palocsay, "Cellular vs process layouts: An analytic investigation of the impact of learning on shop performance," *Omega*, vol. 27, no. 5, pp. 583–592, Oct. 1999.
[31] F. Franceschini and M. Galetto, "An empirical investigation of learning curve composition laws for quality improvement in complex manufacturing plants," *J. Manuf. Technol. Manage.*, vol. 15, no. 7, pp. 687–699, Oct. 2004.
[32] C. Zong, S. Huang, E. Liu, Y. Yao, and S.-Q. Tang, "Nowhere to hide methodology: Application of clustering fault diagnosis in the nuclear power industry," *IEEE Access*, vol. 7, pp. 179864–179879, 2019.
[33] D. Loske and M. Klumpp., "Smart and efficient: Learning curves in manual and human-robot order picking systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 10255–10260, 2020.

YONGMING YAO was born in Yangzhou, Jiangsu, China, in 1987. He received the B.S. degree in communication engineering from the Nanjing University of Posts and Telecommunications, in 2010, and the M.S. degree in computer system architecture from the Xi'an University of Posts and Telecommunications, in 2013. He is currently pursuing the Ph.D. degree in software engineering with Army Engineering University of PLA.

Since 2013, he has been an Assistant Professor with the Software Engineering Department, Tongda College, Nanjing University of Posts and Telecommunications. His research interests include crowdsourced software testing and android permissions detection.

SONG HUANG was born in Huainan, Anhui, China, in 1970. He 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.

CHENG ZONG was born in Yangzhou, Jiangsu, China, in 1986. He received the B.S. degree in electric engineering and the M.S. degree in control engineering from Southeast University, Nanjing, China, in 2008 and 2014, respectively. He is currently pursuing the Ph.D. degree in software engineering with the Army Engineering University of PLA. He is currently working with Jiangsu Nuclear Power Corporation. His research interests include data mining, industrial control, and failure analysis.

ERHU LIU was born in Xuzhou, Jiangsu, China, in 1986. He received the bachelor's and master's degrees from Southeast University, in 2008 and 2013, respectively. He is currently pursuing the Ph.D. degree in software engineering with the Army Engineering University of PLA. His research interests include AI testing and metamorphic testing.

NING CHEN was born in Wuxi, Jiangsu, China, in 1998. She received the bachelor's degree in software engineering (embedded) from the Tongda College, Nanjing University of Posts and Telecommunications, in 2020.