2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT) | 978-1-6654-3757-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/CECIT53797.2021.00037

# Acceptance Testing Optimization Method for Continuous Delivery

HU Peng, Chang Chaowen, MA Yingying
Information Engineering University
Zhengzhou 450001, China
e-mail: hup_xd@163.com

WANG Xiaolin
Jiaxing University
Jiaxing 314001, China
e-mail: wangxiaolin@zjxu.edu.cn

***Abstract*—From agile to DevOps, development methods have been extended from continuous integration to continuous delivery. In the acceptance testing before software delivery, the constantly changing software configuration, real user environment, and user-led testing, all of these extend the delivery time and increase the risk of testing. How to further optimize the test and shorten the delivery cycle are problem that must be considered to realize the end-to-end value flow in DevOps. In the testing optimization research, the methods are mainly for regression testing. There are few researches on acceptance testing, and the methods for the acceptance testing mainly focus on specific scenario, that almost none research consider how to shorten the continuous delivery cycle. Aiming at the characteristics of acceptance testing, this paper proposes an acceptance testing optimization method for continuous delivery. For different levels of test cases, test case selection and prioritization are used to optimize test cases. Firstly, the test suite related to requirements is constructed, and the test cases are selected according to the requirements. Then, the test suites are divided into two levels to prioritize. During the test execution, the use case execution actions are streamlined to shorten execution time of the acceptance testing, meet user needs as soon as possible and achieve rapid value delivery. Finally, the method is applied to actual industrial projects for experiments. The results show that the method can reduce the scale of test cases, shorten the test execution times and improve the efficiency of demand satisfaction.**

***Keywords-testing optimization; acceptance testing; test case selection; test case prioritization***

## I. INTRODUCTION

With the emergence of agile development, DevOps [1] and other modes, software development methods continue improving. At the same time, the testing process and methods have also undergone tremendous changes [2]. In the traditional testing mode, unit testing, integration testing, system testing, and acceptance testing are carried out in sequence according to the software development waterfall model. Therefore, most research on test optimization methods is also carried out on this basis. In the agile mode, the software enters continuous integration and continuous delivery. The test phases are no longer carried out in sequence, but iteratively [3]. Traditional test optimization methods cannot adapt well to the changing test phase. Research is oriented to continuous Optimization strategies under the integration and continuous delivery scenarios, but the current research on optimization methods for different testing phases is relatively lacking.

Acceptance testing is used to verify whether the system could meet the customer needs. It is the final stage of the test [4]. The quality of the acceptance testing directly affects whether the system can be successfully delivered. In the existing research, there are few researches on the optimization of acceptance testing. The existing test case optimization can be divided into coverage-based and error detection rate. Acceptance testing pays more attention to user needs. However, the existing coverage-based optimization methods are mainly oriented to regression testing, and the requirements covered are after conversion. The functional requirements of, the granularity of requirements is small, there is a lot of redundancy, and it does not directly reflect the importance of the needs of users' attention. The operating environment and scenarios also need to be considered in the acceptance testing, which are not covered in the optimization technology for regression testing.

Facing the new development model, in view of the problem that the existing test optimization methods are not suitable for acceptance testing, this paper mainly studies the acceptance testing optimization methods for continuous delivery. The main tasks completed are: (1) According to the new changes in the software development model, analyze the traditional test phase division, clarify the application scenarios of the method in this paper, and abstract the optimization factors of the acceptance testing; (2) Construct the acceptance testing for the acceptance testing link Test optimization model, distinguish two different levels between test suites and test suites, and adopt different optimization methods to ensure both the user-oriented characteristics of acceptance testing and the priority testing of software core functions; (3) Using actual industrial projects, setting The parameter conditions required for the acceptance testing are experimentally evaluated for the method in this paper.

## II. BACKGROUND AND RELATED STUDIES

In iterative development, in order to shorten iteration time and speed up delivery, test-driven development (TDD), behavior-driven development (BDD) and acceptance test-driven development The emergence of development methods such as (ATDD) [5], advance test writing before development, so that development can always target requirements and reduce errors and deviations from requirements.

978-1-6654-3757-8/21/$31.00 ©2021 IEEE
DOI 10.1109/CECIT53797.2021.00037

The existing acceptance testing research mainly focuses on the acceptance test-driven development method. Jeeva et al. [6] conducted research on acceptance testing in agile practice and used the Scrum framework for user acceptance testing. By applying the framework in a large-scale system, the relevant acceptance testing process was defined, so that the efficiency of the system's acceptance testing was obtained. It has been greatly improved, confirming the applicability of the Scrum framework. Maurizio et al. [7] proposed an interactive acceptance testing method for the Internet of Things system, which converts each test scenario into a state machine representing system behavior, and automatically generates corresponding test data according to the graphical user interface of the user operation. implement. This method is mainly for the Internet of Things system, and does not consider the optimization problem of the acceptance testing. Maciel et al. [8] proposed a requirement model-driven acceptance testing method. By adopting the Robot language that supports requirement modeling, the requirements are automatically converted to test cases and test scripts to realize automated acceptance testing. However, this method requires the developer to choose the language as the basis for development and testing from the beginning, which is quite limited. Shin et al. [9] conducted research on the optimization of acceptance testing in the development of cyber-physical systems, taking into account the high environmental dependence and high test risks of cyber-physical systems, and took time budget, hardware damage risk, environmental uncertainty and other factors as constraints, Based on the multi-target search algorithm, prioritize test cases. At the same time, redundant operations between test cases are considered for optimization and streamlining. The experimental evaluation of typical cases in the satellite field confirms the effectiveness of the method for the optimization of the acceptance testing of the cyber-physical system. This method is mainly optimized for the acceptance testing in a specific field, and it is difficult to be applied in practice in other fields. Tonkin et al. [10] aimed at the problem of repeatedly monitoring the operation of different hardware when testing software deployed in the hardware, which consumes a lot of time, and proposed an acceptance testing method that reduces the number of iterations by migrating data between different platforms. This method is mainly optimized for the acceptance testing in a specific environment, only reducing the number of tests, and there is no discussion on how to optimize the test cases. Amos et al. [11] studied the method of reducing the acceptance testing time through test parallelism, and evaluated the relationship between different parallel numbers and system acceptance passes to find the most reasonable test times. Factors such as requirements are not considered, and there is a lack of actual project verification.

## III. ACCEPTANCE TESTING OPTIMIZATION METHOD FOR CONTINUOUS DELIVERY

### A. *Acceptance testing optimization model construction*

In order to implement the acceptance testing optimization, this article classifies the main factors that need to be considered in the test into functional importance, scenario importance, test risk, and test time. Functional importance directly reflects the purpose of software development and is determined by user needs and testing requirements; the importance of scenarios is determined by software operating environment configuration, data input, etc.; testing risks are the main risks faced by testing for different types of software and actual use.

For the optimization of the acceptance testing, mainly by reusing the test cases in the use case library, according to different optimization granularities, the test cases are divided into two levels: test suite and test case for optimization and sorting respectively, as shown in Figure 1. A test suite is a set of test cases for the same function or performance item. The test suite is used to optimize the test according to the importance of the function and risk factors. In the same test suite, different test cases express different input scenarios. Therefore, Optimized in the test suite for the importance of the scenario and the execution time of the use case. Meet the needs of all aspects to the greatest extent, and realize the overall optimization of the acceptance testing.

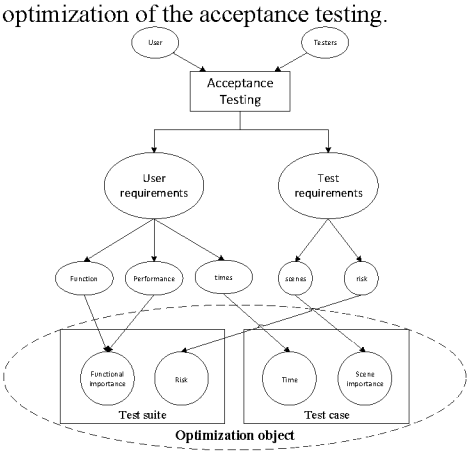


Figure 1. Acceptance testing optimization model for continuous delivery

Acceptance Testing Optimization (ATO) for continuous delivery is mainly divided into three steps. First, select test cases (ATO-RS) based on requirements, construct new test suites, use requirements mapping, and search in the use case library For the test suite associated with the requirement, according to the test target and use case output, search for the test case directly corresponding to the requirement and set the priority to form a new test suite; then prioritize the test cases in the test suite (ATO-TP) , The sorting is based on three factors: priority parameters, scene importance and running time; finally, priority sorting between test suites and execution action reduction (ATO-SP) are carried out, and the importance of requirements is calculated from the importance of function and test risk, and then Prioritize the test suites associated with requirements, and prioritize testing whether the user's more concerned needs are met. When the test is executed, the test cases in the same test suite are streamlined to initialize the actions, and the user needs are verified. Shorten the test time as much as possible and reduce the test risk.

### B. ATO-RS:Test case selection based on requirements

The test suites in the existing test case library are all built based upon functional modules, and the redundancy is too large, and targeted testing cannot be performed according to user needs. It is necessary to build a demand-based test suite to streamline the test cases.

Definition 1. Test case $t$. A test case is the smallest unit that a test can execute, and it should include preconditions, inputs, expected results, and priority. Expressed as a four-tuple as $t(f,l,o,p)$, $f$ is the precondition of the test case, $l$ is the test case input, $o$ is the expected result of the test case, $p$ is the priority of the test case. The priority is set and adjusted according to the needs of the test, and the initial value is set to 0.

Definition 2. Test suite $TS$. A test suite is a set of test cases constructed according to specific test goals and tasks. Therefore, a test suite should include preconditions and target output. A two-tuple is expressed as $TS(F,O)$, and $F$ is the precondition of the test suite. $O$ is the target output of the test suite. In a test suite, the preconditions and expected results of one and at least one test case are consistent with the preconditions and target output of the test suite, to ensure that the test suite can achieve its test goals.

$$TS=\{t_1,t_2,\cdots,\boldsymbol{t_n}\mid \exists \boldsymbol{t_i}, f_i=F \ and \ o_i=O\}$$

In this article, test cases that meet the above conditions are referred to as consistent use cases of the test suite.

Suppose that the user requirement set $R=\{r_1, r_2, \cdots, \boldsymbol{r_n}\}$ of the acceptance testing is set, and there is an existing test case library $AT=\{TS_1,TS_2,\cdots,\boldsymbol{TS_m}\}$, and $TS_i$ is the original test suite. The steps to build a requirement-related test suite are as follows:

1. According to user needs $r_i$, establish a corresponding test suite $TS(r_i)=\{\}$;

2. Find the test suite $TS_j$ that is the same as the requirement target. We extract the consistent use cases of the suite from $TS_j$, add to test suite $TS(r_i)$, and set the test case priority to 1;

3. Take the precondition of test suite $TS_j$ as the target $F_j$, find the test suite $TS_k$ which target is $O_k=F_j$, extract the consistent use case of $TS_k$, add test suite $TS(r_i)$, and set the test case priority to 2;

4. Follow step 3 to continue execution, find new consistent use cases and set the priority of test cases in turn, until there is no test suite target output consistent with the preconditions of the previous test suite, then the test suite corresponding to the requirement is built.

According to the above steps, the corresponding test suites are established for the acceptance testing requirements one by one to form a test suite set associated with the requirement set.

### C. ATO-TP: Optimal sorting of test cases in the test suite

In the constructed requirement-related test suite, the output targets of different test cases are the same, that is, test cases of the same target are tested under different input conditions, and these input conditions represent different environmental scenarios. For example: a functional test suite for providing a service through a web page should include test cases when the service is provided through different browsers, so one browser represents one scenario.

In order to ensure the normal operation of the software functions in each scenario as much as possible, the pre-acceptance testing usually contains most of the foreseeable scenarios. Although this can ensure the software quality and stable operation, in the acceptance testing, run all of them in sequence. The test cases in the scenario may cause the scenarios that are rarely used in some functions have been tested in a limited time, but the important scenarios in some functions have not been tested, and the test cases in the test suite need to be optimized and sorted, Improve the pertinence of user needs and the effectiveness of test execution.

According to the construction process of 3.2, assuming that the requirements are associated with test suite $TS(r_i)=\{t_1,t_2,\cdots,\boldsymbol{t_s},\cdots\}$, for test cases $t_s(f_s,l_s,o_s,p_s)$, $t_{s+i}(f_{s+i},l_{s+i},o_{s+i},p_{s+i})$, if $p_{s+i}=p_s+1$, then $o_{s+i}=f_s$, the test cases are sorted in the order of $\{t_{s+i},t_s\}$, so that the output result of the previous use case is the previous one of the next use case Set conditions, when the use case is executed, the initialization process can be cut to shorten the execution time; if $p_{s+i}=p_s$, it means that the two test cases are the same target test cases in different scenarios, that $l_{s+i}\neq l_s$, $f_{s+i}=f_s$, $o_{s+i}=o_s$, and the test cases need to be performed according to scenario $l$. The most important scenario should be the commonly used scenario, with the most verification times and relatively low risk. Therefore, first select the most important scenario and keep its priority unchanged. The remaining scenes have the same test goals, set their priority to 0, and sort them according to the length of execution time. Tests are performed after the main requirements are tested, which not only guarantees the priority testing of the overall requirements, but also can be tested in a unit of time according to the test needs. Test more scenarios.

### D. ATO-SP: Prioritization of test suites and streamlined execution actions

In order to meet the important requirements as soon as possible, corresponding tests need to be executed according to the priority of the requirements. In 3.2, a one-to-one correspondence between requirements and test suites is realized. Therefore, the importance of each requirement can be calculated according to the functional importance and risk of the impact factors between the test suites, to realize the priority ranking according to the importance of the requirements.

Suppose the importance of requirement $r$ is $V(r)$, which mainly includes functional importance $V(u)$ and software nature importance $V(s)$. Functional importance $V(u)$ represents the direct requirements of users, mainly derived from software requirements specifications, project contracts and other acceptance documents; software nature importance $V(s)$ is mainly determined by the software itself based on software risks, combined with software type, development purpose, etc. For the basic requirements in terms of function and performance, the calculation formula of $V(r)$ can be expressed as:

$$V(r) = \omega_1 \cdot V(u) + \omega_2 \cdot V(s)$$

$$\omega_1 + \omega_2 = 1$$

Among them, $\omega_1$ and $\omega_2$ are the weight ratios of the user's choice importance $V(u)$ and the software nature importance $V(s)$ respectively. Acceptance testing requirements have a relatively large granularity. For users or developers, Analytic Hierarchy Process [12] and Quality Function Deployment [13] are used, and their importance is relatively easy to determine. This article selects the expert method to determine the importance to suit different users. Select requirements and software design requirements, $\omega_1$ and $\omega_2$ values are set according to the actual software.

The test case needs to be initialized before execution to meet the precondition requirements. 3.3 After sorting the test case priority, for any adjacent test case $\{t_s, t_{s+1}\}$ with a priority $p$ greater than 0, there must be $o_{s+1} = f_s$. Therefore, when the same test suite is executed, the initialization action can be streamlined and further shortened according to the priority of the use case The execution time of the acceptance testing.

## IV. EXPERIMENT

Acceptance testing needs to be tested in an actual environment. In order to verify the effectiveness of the Acceptance Testing Optimization (ATO) method proposed in this article, actual industrial projects are selected for experiments. In the experiment, by setting different conditions in the existing use cases of the project, as different choices of users, observe: 1) Under different needs, what is the scale reduction effect of the ATO method for test case selection? 2) Under the same demand, can the ATO method increase the rate of demand satisfaction? 3) What is the overall optimization effect of the ATO method on the acceptance testing?

### A. *Experiment object*

According to the accepted projects, the industrial project SDN_Security is selected as the experimental object of this article. The system mainly realizes the encryption and forwarding of the data stream in the SND network. All requirements and test cases for acceptance testing are extracted from the project, and the granularity of requirements and test cases are classified and combined according to the experiments in this article, and the data information formed is shown in Table 1:

TABLE I. DATA INFORMATION EXTRACTED BY SDN_SECURITY PROJECT ACCEPTANCE TESTING

| Requirement | Test suite | Test case |
|---|---|---|
| 1 | 5 | 9 |
| 2 | 3 | 9 |
| 3 | 4 | 8 |
| 4 | 4 | 6 |
| 5 | 5 | 15 |
| 6 | 4 | 7 |

Among them, requirements 1-5 are functional requirements, and 6 are non-functional requirements. The test suite is composed of test cases for each functional module, and the test cases are large test cases that meet the coarse-grained merger.

### B. *experimental method*

Three sets of comparative experiments are set up in the experiment, and the three phases of the method in this paper are compared and verified.

1. Choose ATO-RS for use cases under different needs

There is a lack of time data in the original acceptance testing. First, perform the use cases corresponding to the requirements in Table 1 in order to supplement the original time data. Then, for each requirement, the ATO-RS method is used to select test cases, and the test suites associated with each requirement are obtained. The data in Table 1 is used as the benchmark data to compare the scale reduction of test cases under different requirements.

2. Prioritization of use cases under the same requirement ATO-TP

The use cases in the test suite are prioritized, and the test cases before sorting are used as a benchmark to compare the rate of satisfaction before and after sorting. At the same time, in order to compare the optimization effect of the method in this article on the acceptance testing, the classic priority sorting Total and Additional [14,15] methods are added Compare. During the experiment, take the average of the corresponding data before and after sorting under different requirements, as the result of the experiment for comparison.

3. The overall optimization of the acceptance testing to execute ATO-SP

Taking the original acceptance testing (Initial Test, I-Test) execution as a benchmark, according to the requirement priority sorting method, the test suite set is sorted as a whole and executed, and the optimization effect of the ATO-SP method in terms of execution time and importance of requirements is compared.

### C. *Evaluation indicators*

In the experiment, according to the optimization method and experiment purpose at different stages, the corresponding evaluation index is selected.

（1）*APBC* [16]

This indicator is used to measure the coverage rate of program modules by different sorts of test cases, and its calculation formula [17] is:

$$APBC = 1 - \frac{TB_1 + TB_2 + \ldots + TB_m}{mn} + \frac{1}{2n}$$

Among them, $m$ represents the number of program modules, $n$ represents the number of test cases, and $TB_i$ represents the order in which the first module is covered by the test cases for the first time. In this paper, the test suite corresponding to the requirement is used to represent the program module, that is, $APBC$ is used to measure the coverage rate of the test suite that meets the requirements by different sorts.

（2）Test case scale reduction rate $F$ [18]

The test case scale reduction rate is used to measure the degree of use case reduction. The calculation formula is:

$$F = \left(1 - \frac{|T'|}{|T|}\right) \times 100\%$$

$T$ and $T'$ respectively represent the set of test cases before and after optimization. Although use case reduction can reduce the scale of use cases that need to be executed, it may also cause problems such as reduced coverage. Therefore, it is also necessary to consider the impact of use case reduction on coverage. In the acceptance testing studied in this paper, the coverage rate is the coverage of user requirements. The proposed optimization method selects test cases based on the requirements. Therefore, the coverage rate of the requirements can reach 100%, and the reduction rate $F$ is no longer considered in the experiment. The impact of demand coverage.

### D. *Experimental results and analysis*

#### *1) Choice of test cases under different requirements*

From the test case selection results shown in Table 2, it can be seen that using the ATO-RS method for use case selection of functional requirements will reduce the scale of test cases to a certain extent, while the ATO-RS method has no effect on non-functional requirements.

TABLE II. TEST CASE SELECTION RESULTS UNDER DIFFERENT REQUIREMENTS

| Requirement | Pre-selection test case | Test case after selection | $F$ % |
|---|---|---|---|
| 1 | 9 | 5 | 44.44 |
| 2 | 9 | 3 | 66.67 |
| 3 | 8 | 5 | 37.5 |
| 4 | 6 | 5 | 16.67 |
| 5 | 15 | 6 | 60 |
| 6 | 7 | 7 | 0 |

Figure 2 shows the comparison with the original test case after the scale of each requirement test case is reduced, where Mean_All represents the average reduction rate corresponding to the overall demand, and Mean_Function represents the average reduction rate under the functional requirements. Although the ATO-RS method cannot achieve the reduction effect on non-functional requirements, the overall acceptance testing shows that the selected test cases account for 62.4% of the original test cases, and the overall test case scale reduction effect is obvious. For functional requirements, the selected test cases only accounted for 54.9% of the original use cases, and the reduction effect is even more obvious. Therefore, in actual application, the ATO-RS method can be used to select test cases for the acceptance testing of the functional project.

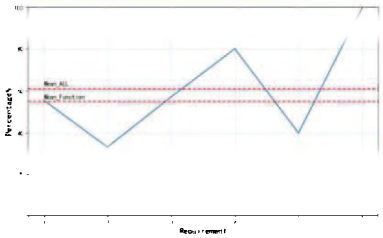

Figure 2. Comparison of test cases before and after reduction

#### *2) Prioritization of test cases under the same requirement*

In this stage, the requirements in the same test suite are compared with the rate of satisfaction. The original test in the project did not use an optimization strategy. In the experiment, the random sorting method was used to perform 15 sets of average values for the test cases before ATO-TP sorting, as the order of the original test cases. Figure 3 shows the comparison of 4 different methods. It can be seen from the figure that ATO-TP is superior to the original project test and the Total and Additional methods in terms of average demand coverage rate and stability, although optimization is not used in the original test, a higher situation may occur, but this kind of uncertainty needs to be avoided as much as possible in the acceptance testing. Analyze the two methods of Total and Additional, and sort the coverage of requirements based on test cases. In the acceptance testing, the granularity of test cases is larger, and different test cases meet the same requirement less frequently. From the experimental results, it can also be seen that the two This method is not ideal for sorting coarse-grained test cases.

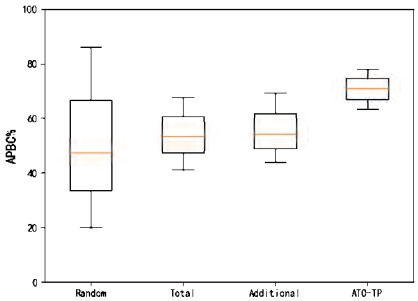


Figure 3. Comparison of $APBC$ under different sorting methods

#### *3) Optimized overall execution of acceptance testing*

Taking the original acceptance test execution as a benchmark, according to the requirement priority sorting

method, the test suite set is sorted as a whole and then executed, and the optimization effect of the ATO-SP method in terms of execution time and importance of requirements is compared.

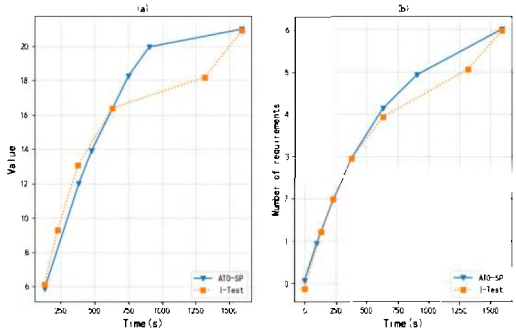


Figure 4. Sorting effect under different selection situations

The ATO-SP method is sorted according to the importance of requirements. Figure 4 shows the importance of function and the length of time as the basis of importance of requirements, indicating that users are more concerned about the function or the time to meet the requirements. Due to the limited amount of demand data, the importance of changes over time and the number of demands before and after sorting are fitted to obtain the curve changes shown in the figure. In figure (a), as time goes by, under the same time, ATO-SP The importance of method execution use cases is higher. In Figure (b), the ATO-SP method can meet more needs in the same time. Therefore, the ATO-SP method can adapt to different user choices for overall optimization.

## V. CONCLUSIONS

This paper studies the acceptance test in the continuous delivery scenario, adopts the method of combining test case selection and prioritization, and adopts the optimization strategies of ATO-RS and ATO-TP at different levels, and adopts the ATO-SP method during execution. Streamline actions to achieve overall optimization of acceptance testing. Among them, the ATO-RS method selects test cases based on the requirements and constructs a test suite related to the requirements; the ATO-TP method prioritizes the test suites according to the priority parameters, test risks and time, so as to meet the test requirements as soon as possible; ATO -SP method prioritizes the test suites according to the user's choice and the importance of the requirements, and during the test execution, the initialization action is streamlined, which further reduces the acceptance test execution time. Experiments in small industrial projects show that the method proposed in this paper can improve the efficiency of acceptance testing requirements and reduce the scale of test cases and test execution time.

## ACKNOWLEDGMENT

This research was supported by Natural Science Foundation of Zhejiang Province under grant No. LQ22F020004.

## REFERENCES

[1] Akbar M A, Mahmood S, Shafiq M, et al. Identification and prioritization of DevOps success factors using fuzzy-AHP approach[J]. Soft Computing, 2020.

[2] Angara J, Prasad S, Sridevi G. The Factors Driving Testing in DevOps Setting- A Systematic Literature Survey[J]. Indian Journal of Science & Technology, 2017, 9(48).

[3] Hellmann T D, Chokshi A, Abad Z, et al. Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences[C]. Agile Conference. IEEE, 2013.

[4] Mussa M, Khendek F. Acceptance Test Optimization[C]. System Analysis & Modeling: Models & Reusability. 2014.

[5] MM Moe. Comparative Study of Test-Driven Development TDD, Behavior-Driven Development BDD and Acceptance Test–Driven Development ATDD[J]. International Journal of Trend in Scientific Research and Development, 2019:231-234.

[6] Jeeva P, Perera I, Bandara D. Applying agile practices to avoid chaos in User Acceptance Testing: A case study[C]. Moratuwa Engineering Research Conference (MERCon), 2016.

[7] Maurizio, Leotta, Diego, et al. An acceptance testing approach for Internet of Things systems[J]. Iet Software, 2018.

[8] Maciel D, Paiva A, Silva A R D. From Requirements to Automated Acceptance Tests of Interactive Apps: An Integrated Model-based Testing Approach[C]. International Conference on Evaluation of Novel Approaches to Software Engineering, 2019.

[9] Shin S Y, Nejati S, Sabetzadeh M, et al. Test Case Prioritization for Acceptance Testing of Cyber Physical Systems: A Multi-Objective Search-Based Approach[C]. ACM SIGSOFT International Symposium on Software Testing and Analysis, 2018:49-60.

[10] Tonkin E L , Nieto M P , Bi H , et al. Towards a Methodology for Acceptance Testing and Validation of Monitoring Bodyworn Devices[C]. IEEE International Conference on Pervasive Computing and Communications Workshops. IEEE, 2020.

[11] Amos E, Gera. Acceptance testing for a planar array[J]. Communications in Statistics - Simulation and Computation, 2020, 49(5): 1198-1209.

[12] Ruhe G, Greer D. Quantitative studies in software release planning under risk and resource constraints. In: Proc. of the 2003 Int'lSymp. on Empirical Software Engineering (ISESE 2003). Los Alamitos: IEEE Computer Society Press, 2003. 262−271.

[13] Wei S, Shu-Juan L I, Yan L I. Confirming weight of voice of the customer in QFD using the method of AHP[J]. Machinery Design & Manufacture, 2005.

[14] Rothermel G, Untch R H, Chu C, et al. Prioritizing Test Cases For Regression Testing[J]. IEEE Transactions on Software Engineering, 2000, 27(5):102-112.

[15] Sharma N, Sujata, Purohit G N. Test case prioritization techniques "an empirical study"[C]. 2014 International Conference on High Performance Computing and Applications (ICHPCA). IEEE, 2015.

[16] Li Z, Harman M, Hierons R M, Search Algorithms for Regression Test Case Prioritization[J]. IEEE Transactions on Software Engineering, 2007, 33(4):225-237.

[17] Chen X, Chen J H , Xiao-Lin J U , et al. Survey of Test Case Prioritization Techniques for Regression Testing[J]. Journal of Software, 2013.

[18] Hu P, Chang C W, Zhu X W, et al. Regression Testing Optimization Method for Continuous Integration[J]. Application Research of Computers, 2021.