Test Case Prioritization based on Requirement Correlations*

Tingting Ma[1,2], Hongwei Zeng[1], Xiaolin Wang[1,2]
[1]School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
[2]Shanghai Key Laboratory of Computer Software Evaluating & Testing, Shanghai 201112, China
tingtingyuli_1992@163.com
{zenghongwei, wangxiaolin}@shu.edu.cn

***Abstract*—Test case prioritization technique aims to improve test efficiency rate by sorting test cases according to some specific criteria. Requirements play an important role throughout software testing. This paper proposes a test case prioritization method based on requirement correlations. Prioritization of requirements is defined by the users and the developers. This technique focuses on requirements with detected faults after the last regression testing. By readjusting prioritization of fault-related requirements, it can optimize the order of test cases. Experimental results show that this technique exactly contributes to achieving high testing efficiency.**

Keywords—Test case prioritization; Requirement correlations; Fault detection rate; Regression Testing

I. Introduction

With the development of computer science and technology, software industry is developing rapidly, and software tends to be modeled, open, rational and global. As software evolution process makes software more complicated and software development process is more difficult to control, software testing plays an important role in the software development cycle in order to ensure the quality of software. Software testing runs through the whole process of the software development. Regression testing, as a component of the software life cycle, has a large proportion of the workload in the whole process of software testing. Numerous regression testing are needed in the various stages of the software development. On account of the increasing scale and complexity of software system, it consumes a considerable number of resources and time, which makes it impractical to execute all the test cases in every regression testing.

For reducing the testing costs and improving testing efficiency, many researchers presented a variety of optimization methods, such as test case selection, test case prioritization and test suite augmentation [1, 2]. The research of test case prioritization (TCP) can be traced back to the paper published in 1997 by Wong et al [3]. They prioritized test cases for regression testing and did empirical studies to verify the effectiveness of their method. TCP technique tries to optimize the order of the test cases so as to improve regression testing efficiency. In regression testing, various kinds of factors, which affect the priority ordering of test cases, can serve as prioritization criteria. Existing prioritization criteria consists of requirement, coverage (statement coverage or function coverage), historical information and cost, etc [4-9]. Requirement analysis is an indispensable stage of the software development cycle. Requirement [10-14], as a significant factor influencing prioritization of test cases, has been considered in many papers to adjust the prioritization of test cases.

Xiaofang Zhang et al. [10] proposed test case prioritization based on varying testing requirements priorities and test case costs. They considered three influence factors of requirement including requirement volatility, fault proneness of the requirement and customer-assigned priority. They brought forward the metric to evaluate the rate of "units-of-testing-requirement-priority-satisfied-per-unit-test-case-cost".

R. Krishnamoorthi et al. [11] presented the test case prioritization called Prioritization of Requirement for Test (PORT 6). They combined six factors of requirement to prioritize the system test cases including customer priority, changes in requirement, implementation complexity, completeness, traceability and fault impact. Their prioritization technique is validated with lots of empirical research, and it improved the rate of severe fault detection.

Srikanth H et al. [12] took four factors of requirement into consideration to improve test efficiency through system test prioritization. Unlike R. Krishnamoorthi et al., their factors contained customer priority, implementation complexity, fault proneness and requirements volatility.

The prioritization method is proposed by R. Kavitha et al. [13] had four requirement factors: customer assigned priority of requirements, developer-perceived code implementation complexity, changes in requirements and fault impact. Although their factors are similar with Srikanth H et al., their practical algorithms are different.

In this paper, test case prioritization based on requirement correlations is presented. Sorting test cases is founded on requirement correlations, and priorities of test cases are dynamically adjusted in the regression testing execution. The rest of this paper is arranged as followed. Section 2 describes related knowledge about test case prioritization. The proposed method is detailed in section 3. Section 4 is case studies to verify the effective of this method. The conclusions of this paper and the outlook of future work are described in section 5.

*This work is supported by National Natural Science Foundation of China (NSFC) under grant No.61170044 and No. 61572306.

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 16:10:15 UTC from IEEE Xplore. Restrictions apply.

II. Test Case Prioritization

Test case prioritization is designed to make a readjustment on test cases according to some given criteria in order to increase software testing efficiency. Different test cases have different abilities to detect faults and to cover requirements. By a certain ordering criterion, it will increase priority of test cases which have strong detection ability and improve fault detection rate or requirement coverage rate. For TCP problem, Elbaum et al. [15] formally provided its formal definition as follows:

Given: T, a test suite; PT, the set of permutations of T; f, a function from PT to the real numbers.

Problem: Find $T^{'} \in PT$ such that

$$(\forall T^{''})(T^{''} \in PT)(T^{''} \neq T^{'})[f(T^{'}) \geq f(T^{''})].$$

Here, PT is the set of all possible execution orders of all test cases in T, and f, whose input is specific execution order, is an objective function that yields an award value. The output of f is related to scheduling target: the larger the value of f is, the better the sorting efficiency is.

III. Prioritization Method

Requirement has a great effect on resorting test cases, so at this point, this paper proposes requirement based dynamical adjustment test case prioritization algorithm. This section first describes requirement prioritization initialization, and then explains some definitions about requirement correlations, and finally designs dynamic adjusting prioritization process.

A. Requirement Prioritization Initialization

Requirement priority is influenced by several factors. In this context, initial requirement priority is mainly decided by customer-perceived priority (CP) [16] and developer-perceived priority (DP) [8].

- Customer-perceived Priority (CP) is evaluated by the customer to show the importance of each requirement to customer in accordance with functional requirements specification. The customer assigns a value to each requirement ranging from 1 to 10 with 10 representing the highest customer-perceived priority. The customers' satisfaction is the most important goal for all works through the software development. Accordingly, the requirement, which has high value of customer-perceived priority, should be executed early to increase customer satisfaction.
- Developer-perceived Priority (DP) is designated by the developer to indicate how difficult to implement each functional requirement with code programming. The developer assigns a value to each requirement ranging from 1 to 10 with 10 representing the highest developer-perceived priority. More difficult a requirement is completed, higher it should be valued. So, the requirement, which has high value of developer-perceived priority, should be executed early.

Based on CP and DP, the calculating formula for initial requirement priority (RP) of the i th requirement is as follows:

$$RP_i = CP_i + DP_i \qquad (1)$$

where customer-perceived priority and developer-perceived priority are of equal importance.

B. Requirement Correlations

Individual requirements are generally not independent but interrelated. There may be a dependency relationship between one requirement and the others, such as the inclusion relation. Functional requirements of the modules often contain a number of sub-functional requirements. Only all of sub-functional requirements are tested to pass, a functional requirement is considered to pass the test.

Definition 1: Given a functional requirement including several sub-functional requirements, it is called father-son relationship between the functional requirement and sub-functional requirement. It is called sibling relationship between two sub-functional requirements.

In regression testing, the functional requirement with detected faults has larger possibility to find faults again than others. This paper concentrates on functional requirement with detected faults. The criteria for Requirement Correlation (RC) are as follows:

- The functional requirement node with detected faults is assigned the highest priority.

  Reasoning: After modifying codes, it may change the functionality of this functional requirement node so that new faults will be introduced.
- The paternal dependent requirement node of the functional requirement node with detected faults is assigned the higher priority.

  Reasoning: When all tests of all requirement nodes that paternal requirement node invokes pass, the paternal requirement node is believed to be passed.
- The sibling requirement node of the functional requirement node with detected faults is assigned the lower priority.

  Reasoning: The sibling requirement node may have the same or similar function with the functional requirement node with detected faults.
- The child dependent requirement node of the functional requirement node with detected faults is assigned the lowest priority.

  Reasoning: After modifying codes, if the functionality of this functional requirement node with detected faults remains unchanged, it will not have a great impact on the child dependent requirement node.

C. Test Case Prioritiziation Method

Each test case has its own weight. Test case prioritization reorders test cases in a descending order by weights of test

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 16:10:15 UTC from IEEE Xplore. Restrictions apply.

cases to improve testing efficiency. Before getting weights of the test cases, it needs to setup a functionality dependency graph (FDG) [17] according to the functional requirement specification. On the basis of FDG, The dependency relationship between two requirements is called path, and each path is a test case. The weight of a test case is the cumulative sum of priority of the requirements included. Firstly, test case with the highest priority is selected to execute, and then priority of test case not being performed is dynamically adjusted according as the rule of requirement correlations. Specific TCP algorithm flow chart is shown in Fig.1.

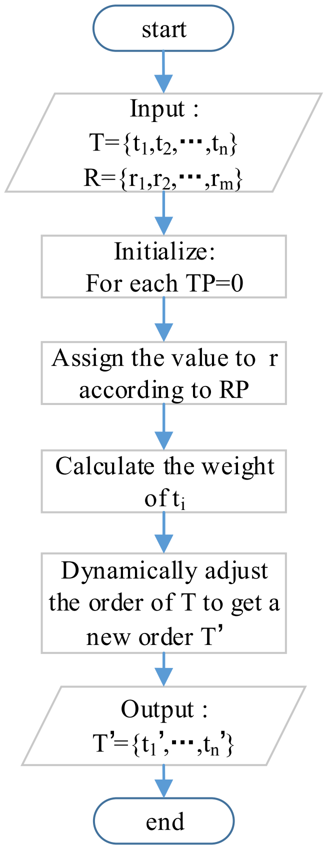
start
Input :
T={t1,t2,⋯,tn}
R={r1,r2,⋯,rm}
Initialize:
For each TP=0
Assign the value to r according to RP
Calculate the weight of ti
Dynamically adjust the order of T to get a new order T'
Output :
T'={t1',⋯,tn'}
end

Fig.1. TCP algorithm flow chart

Let $T=\{t_1,t_2,...,t_n\}$ for a test suite and $R=\{r_1,r_2,...,r_m\}$ for a set of requirements. For every test case $t\in T$, there is at least one requirement r in R that satisfies t. Similarly, for every requirement $r\in R$, there is at least one test case t in T that satisfies r. $S(T,R)_{n\times m}$ denotes to describe the relation of T and R : $S(t_i,r_j)$ = RP_j ,if t_i satisfies r_j ; $S(t_i,r_j)$ =0,if t_i doesn't satisfy r_j . Let TP for the weight of a test case: $TP=\{TP_1,TP_2,...,TP_n\}$. The initial value of TP is set to 0. The algorithm of computing the weight of t_i is as follows.

Algorithm 1 Computing the weight of t_i based on the satisfiability relation of test cases and requirements

```
1. for ( i=1; i<=n; i++ )
2.    for ( j=1; j<=m; j++ )
3.    {
4.          $TP_i += S(t_i,r_j)$;
5.    }
```

After the first-time regression testing, it's available to get requirements that have found faults. The next stage is to adjust the value of requirement priority using the proposed method. Let F_j for the j th requirement finding faults: if r_j finds faults, then $F_j=1$. Functional requirement dependent graph is described as a directed acyclic graph G=(V,E): let $V=\{r_1,r_2,...,r_m\}$ as a set of requirement nodes and $E=\{e_1,e_2,...,e_s\}$ as the dependency relationship between two requirements. Below is the algorithm of adjusting the priority of requirements.

Algorithm 2 Adjusting requirement priority

```
1. for each $r_i$
2.    if ( $F_i$ =1 )
3.        then give $RP_i$ the highest priority;
5.        if ( $r_i$ has father node(s) )
6.            then give $RP_{j\_father}$ higher priority;
                                    //all father nodes of $r_i$
7.        endif
8.        if ( $r_i$ has sibling node(s) )
9.            then give $RP_{i\_brother}$ lower priority;
                                    //all sibling nodes of $r_i$
10.       endif
11.       if ( $r_i$ has sub-node(s) )
12.           then give $RP_{i\_sub}$ the lowest priority;
                                    //all sub-nodes of $r_i$
13        endif
14.   endif
15. endfor
```

The assigned value of requirement priority depends on actual conditions of the experiment by the criterion of requirement correlations. Here the importance of requirement priority can be assigned to 5,4,3,2 from high to low.

Algorithm 1 is applied to get test priorities of test cases. It needs to sort test cases in a descending order according to test priorities of test cases. When testing detects faults for a functional requirement, other related requirements may find related or dependent faults according to requirement

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 16:10:15 UTC from IEEE Xplore. Restrictions apply.

correlations. Therefore it is necessary to increase the priority of functional requirements related to functional requirement with detected faults. Accordingly, it will increase the priority of test cases that satisfy functional requirements with detected faults. There are two methods to adjust test prioritization: total adjusting prioritization and additional adjusting prioritization. Total adjusting prioritization is defined to optimize test cases according to descending order of test priorities in order to execute the test cases with higher priority earlier. Additional adjusting prioritization is used to dynamically sort test cases. The executing process of dynamical adjusting prioritization is described as follows:

Step1: set a backup to a set of requirements.

Step2: if requirements are not fully covered, then calculate the priorities of test cases according to Algorithm 1.

Step3: select a test case that has the highest priority.

Step4: remove selected test case from test cases and requirements satisfied by selected test case from a set of requirements, and recalculate the priorities of test cases.

Step5: repeat Step2 and Step4 until all the requirements are performed.

Step6: if requirements are fully covered, then make set of requirements as a backup of requirements in Step1.

Step7: repeat the above steps until all the test cases are performed.

In the steps of the regression testing, Step3 and Step4 are the most important. Step3 and Step4 are the process of additional adjusting test prioritization.

The algorithm of additional adjusting test prioritization is shown as follows.

Algorithm 3 Additional adjusting test prioritization

Input: $T=\{t_1,t_2,...,t_n\}$, $R=\{r_1,r_2,...,r_m\}$, $S(T,R)_{n\times m}$

Output: $T^{'}$

```
1. $T^{'}=\phi$
2. $R^{'}=R$                    // set a backup to requirements
3. while($T\neq\phi$) do
4.     while($R^{'}\neq\phi$) do
5.         calculate $TP_i$ of each test case in $T$ by Algorithm 1
6.         select one test case $t_k$ with the highest priority
7.         $R^{'}=R^{'}-R^{'}(t_k)$;
                       //requirements satisfied by selected test case
8.         $T=T-\{t_k\}$;
9.         $T^{'}=T^{'}+\{t_k\}$
10.    endwhile
11.    $R^{'}=R$
12. endwhile
```

IV. Experiment

A. A Case Study

An example is applicable to explain the process of the proposed method. Assume a test suite $T=\{t_1,t_2,t_3,t_4\}$ and a set of requirement priorities $R=\{r_1,r_2,r_3,r_4,r_5,r_6,r_7,r_8\}$. Suppose r_2, r_3 and r_4 depend on r_1; r_5 and r_6 depend on r_3; r_7 depends on r_4; r_8 depends on r_5. FDG is shown in Fig.2.

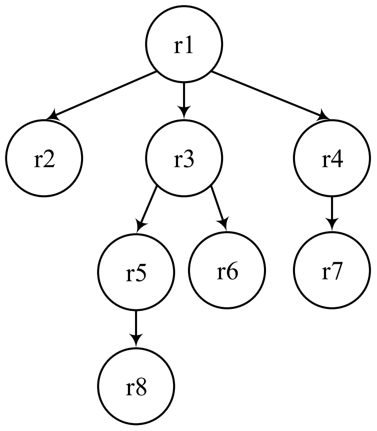

Fig.2 FDG of a case study

Initial priority of requirements is calculated according to Equation (1). Table I describes initial priority of requirements.

TABLE I. INITIAL PRIORITY OF R

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
CP	10	5	7	8	6	4	3	2
DP	10	7	9	7	4	5	3	3
RP	20	12	16	17	10	8	6	5

Suppose: r_5 and r_7 have detected faults after the first regression testing. Add 5 to r_5 and r_7; add 4 to r_3; add 3 to r_6; add 2 to r_8; add 5 to r_7; add 4 to r_4. Adjusted RP of R is shown in table II.

TABLE II. ADJUSTED RP OF R

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
RP	20	12	20	21	15	11	11	7

From algorithm 1, the TP of each test case is calculated as in Table III.

TABLE III. DETAILS OF $S_{4\times 8}$

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
t_1	20	12	0	0	0	0	0	7
t_2	20	0	20	0	15	0	0	7

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 16:10:15 UTC from IEEE Xplore. Restrictions apply.

t_3	20	0	20	0	0	11	0	0
t_4	20	0	0	21	0	0	11	0

TABLE IV. TP OF EACH TEST CASE

T	Satisfies R	TP
t_1	r_1, r_2, r_8	32
t_2	r_1, r_3, r_5, r_8	62
t_3	r_1, r_3, r_6	51
t_4	r_1, r_4, r_7	52

From Table IV, TP of test case t_2 is the largest, so t_2 is executed first. According to algorithm 3, the ordering of TCP is $\{t_2, t_4, t_1, t_3\}$.

B. Effeactive Measure

The purpose of TCP is to reorder test cases to detect faults as much as possible. To measure the rate of fault detection, Rothermel et al. first put forward evaluation metrics called APFD(Average Percentage of Faults Detection)[15]. Given execution order of test cases, APFD can calculate average cumulative proportion of faults detected during execution of the test cases. Elbaum et al. gave a formula to compute APFD as follow:

$$APFD = 1 - \frac{TF_1 + TF_2 + \ldots + TF_m}{nm} + \frac{1}{2n} \quad (2)$$

Here, T is a test suite including n test cases and m faults. Suppose a test case execution order, TF_i indicates the order of test case which detects the i th fault in the execution order. Equation (2) shows that the value of APFD ranges from 0 percent to 100 percent. Higher the value is, faster the fault detection rate is.

In practical experiments, many factors need to be taken into account, such as cost of each test case and the severity of the faults. In this experiment, cost of each test case and the severity of the faults are not considered or deemed to be equal. There are two faults in the case study and four test cases in the test suite. That t_2 and t_4 can detect faults is concluded from the fact that r_5 and r_7 can detect faults, t_2 satisfys r_5 and t_4 satisfys r_7. In the next regression testing, faults are still the same ones. Corresponding Relationship of test cases and faults is shown in Table V.

TABLE V. CORRESPONDING RELATIONSHIP

	F1	F2
t_1		
t_2	√	
t_3		
t_4		√

APFD metrics are used for evaluating the rate of fault detection. The APFD value of unsorted $\{t_1, t_2, t_3, t_4\}$ is calculated as below and shown in Fig.3.

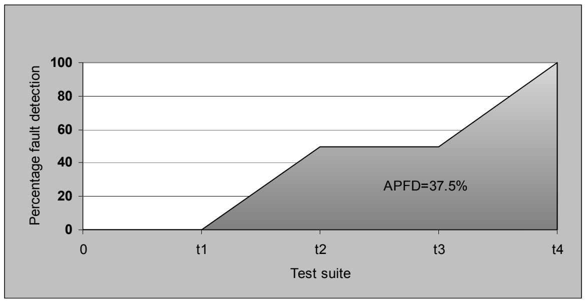

Fig.3 APFD value of the unsorted test cases

The APFD value of sorted $\{t_2, t_4, t_1, t_3\}$ by the proposed method is calculated as below and shown in Fig.4.

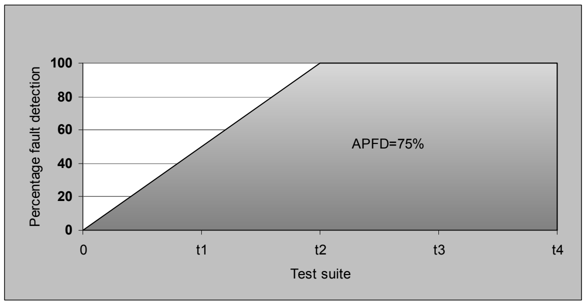

Fig.4 APFD value of the sorted test cases

Comparing to Fig.3, Fig.4 shows that the proposed method is verified to be able to improve test efficiency under certain conditions.

C. Industrial Case Study

Here's an example of an industrial case study with a web application, which has 59 functional requirements, 42 test cases and 18 faults. The experimental results and analysis are shown below. Fig.5 is AFPD value of the unsorted test cases.

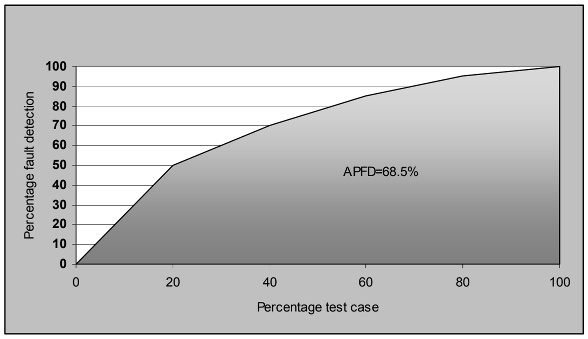

Fig.5 APFD value of the unsorted test cases

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 16:10:15 UTC from IEEE Xplore. Restrictions apply.

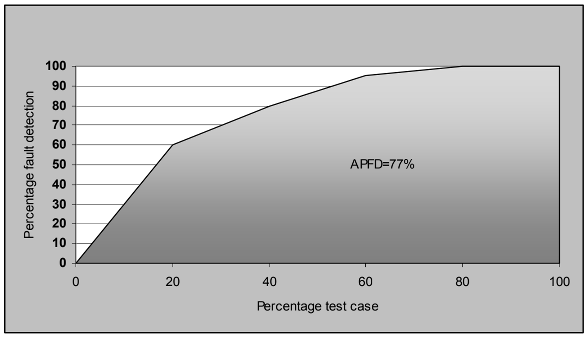

Fig.6 APFD value of the sorted test cases

Comparing Fig.5 with Fig.6, the proposed method is more effective than the unsorted test cases. The result of increasing requirement priority is to improve test priority. This method is tested on the system testing, but it's not applicable to unit test.

V. SUMMARIZATION AND PROSPECT

A test case prioritization method based on requirement correlations is proposed in this paper. The proposed method first adjusts requirement priority, then calculates the weight of each test case priority, finally sorts test cases dynamically according to the weight of test cases. The proposed method is verified to be efficient on a small experiment example, and the results show that the proposed method is better than no sorting method. In this paper, the severity of faults is not considered, so it will be a research focus to improve test efficiency in the future work.

REFERENCES

[1] Yoo S, Harman M. Regression testing minimization, selection and prioritization: A survey. Software Testing, Verification & Reliability, 2012,22(2):67−120. [doi: 10.1002/stvr.430]

[2] Harrold M, Orso A. Retesting software during development and maintenance. In: Proc. of the Frontiers of Software Maintenance. IEEE Press, 2008. 99−108. [doi: 10.1109/FOSM.2008.4659253]

[3] Wong W, Horgan J, London S, Agrawal H. A study of effective regression testing in practice. In: Proc. Of the Int'l Symp. On Software Reliability Engineering. IEEE Press, 1997. 264−274. [doi: 10.1109/ISSRE.1997.630875].

[4] Rothermel G, Untch RJ, Chu C. Prioritizing test cases for regression testing. IEEE Trans. on Software Engineering, 2001,27(10): 929−948. [doi: 10.1109/32.962562]

[5] Rothermel G, Untch RH, Chu C, Harrold MJ. Test case prioritization: An empirical study. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 1999. 179−188. [doi: 10.1109/ICSM.1999.792604]

[6] Qu B, Nie CH, Xu BW. Test case prioritization based on test suite design information. Chinese Journal of Computers, 2008,31(3): 431−439 (in Chinese with English abstract).

[7] Chen X, Chen JH, Ju XL, Gu Q. Survey of Test Case Prioritization Techniques for Regression Testing. Journal of Software. 2013,24(8): 1695-1712(in Chinese).

[8] Wang X, Zeng H. Dynamic test case prioritization based on multi-objective[C]//Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on. IEEE, 2014: 1-6.

[9] Kim J M, Porter A. A history-based test prioritization technique for regression testing in resource constrained environments[C]//Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on. IEEE, 2002: 119-129

[10] Zhang X, Nie C, Xu B,et al. Test Case Prioritization based on Varying Testing Requirement Priorities and Test Case Costs[C]//Quality Software,2007. QSIC'07. Seventh International Conference on. IEEE, 2007: 15-24.

[11] R. Krishnamoorthi, S.A. Sahaaya Arul Mary. Factor oriented requirement coverage based system test case prioritization of new and regression test cases[J]. Information and Software Technology. 2009,51(4): 799-808

[12] Srikanth H, Banerjee S. Improving test efficiency through system test prioritization[J]. Journal of Systems and Software, 2012, 85(5): 1176-1187.

[13] R Kavitha, VR Kavitha, NS Kumar. Requirement based test case prioritization[J]. Communication Control & Computing Technologies International Conference on IEEE, 2010: 826-829

[14] Garg D, Datta A, French T. A Two-Level Prioritization Approach for Regression Testing of Web Applications[C]//Software Engineering Conference (APSEC), 2012 19th Asia-Pacific. IEEE, 2012, 2: 150-153.

[15] Elbaum S, Malishevsky AG, Rothermel G. Prioritizing test cases for regression testing. In: Proc. of the Int'l Symp. on Software Testing and Analysis. ACM Press, 2000. 102−112. [doi: 10.1145/347324.348910]

[16] Srikanth H, Williams L, Osborne J. System test case prioritization of new and regression test cases[C]//Empirical Software Engineering, 2005. 2005 International Symposium on. IEEE, 2005: 10 pp.

[17] T.Zimmermann and N.Nagappan, Predicting subsystem failures using dependency graph complexities, in Proceedings of the 18th IEEE International Symposium on Software Reliability. ISSRE '07, 2007: 227-236.

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 16:10:15 UTC from IEEE Xplore. Restrictions apply.