ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022

Prioritization of Dynamic Test Cases Based on Historical Data for Use in Regression Testing of Requirement Properties

Appari pavan kalyan[1], Dr.Harsh Pratap Singh [2], Dr. B.Kavitha Rani[3]

[1]*Research Scholar, Dept. of Computer Science and Engineering, Sri Satya Sai University of Technology and Medical Sciences, Sehore Bhopal-Indore Road, Madhya Pradesh, India.*
[2]*Research Guide, Dept. of Computer Science and Engineering, Sri Satya Sai University of Technology and Medical Sciences, Sehore Bhopal-Indore Road, Madhya Pradesh, India.*
[3]*Research Co-Guide,* Professor, in Dept. of CSE, CMR Technical Campus, Hyderabad

Abstract

The process of regression testing is vital, but it is also quite expensive and time-consuming to carry out. Because there are only so many resources available in practise, the prioritising of test cases places an emphasis on the acceleration of the testing process. However, conventional strategies for prioritising test cases place an emphasis primarily on one-time testing and do not take into account the massive amounts of historical data provided by regression testing. Within the scope of this work, an approach is proposed for ranking test cases by using historical data. The requirements play a vital role in the process of testing; the priorities of test cases are initialised based on the requirements' priorities in our history-based method, and they are afterwards determined dynamically according to the historical data in regression testing. In order to assess the effectiveness of our methodology, we will be carrying out an empirical study on a real-world application. The findings of our experiments demonstrate an improvement in performance for the strategy that we have proposed by employing measures of the "Average Percentage of Faults Detected and the Fault Detection Rate".

Keywords: *dynamic, prioritization, regression*

1. INTRODUCTION

To make great software engineering (SE) decisions, you must be aware of the business repercussions. The majority of SE research is based on value-neutral environments, in which all software creations are given equal importance. Value-based software engineering (VBSE) considers value when designing software concepts and processes. Barry Boehm defines VBSE as "the explicit concern with value issues in the application of science and mathematics". Extreme programming, pair programming, and lean software development dominated early agile software development research. These are three agile software development buzzwords. This pattern is changing, with developed features and continual value delivery becoming the focus. Trends include continuous value delivery and close interaction between business and technical teams. Supposedly, software quality is value-based. Software customers are often concerned with the value software solutions provide to their organisations. Software improves customers' businesses in many ways. Examine expenses and advantages. It's important to know the value clients expect from the program's quality. Software testing is becoming more important to match customer expectations for the

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

program's value. Software testing is an important and expensive component of the SDLC, consuming 40 to 50% of the budget.

Complexity, size, and support for real-time organisations have made software testing a must. Software is growing and enabling real-time enterprises. All code statements, requirements, use cases, circumstances, techniques, and scenarios are treated equally in software testing research. Similar to other software development phases. Software testing jobs that don't add value and have a low ROI are widespread (ROI). Software testing might cost $300 billion globally. Value-neutral testing is testing that is independent of the product's commercial goals [9]. Value-based testing was suggested as a solution.

Value-based testing [8] evaluates software solutions to see if they can better match testing resources to project objectives. Value-based testing requires integrating internal testing objectives with company objectives and customer expectations. The focus is on delivering customer value, not validating code against a list of requirements. According to, value-based testing had a greater return on investment (ROI) of 1.74 with 40% of the most valuable test cases, but value-neutral testing only produced a ROI of 1.22 with 100% of the tests. This means testing resources should help the customer's business. During testing, both customer expectations and stated specifications should be followed. As a result, testing methodologies must focus on business value. VBSE includes value-based verification and validation.

Boehm estimates that testing accounts for 50% of the $1 trillion yearly cost of software. If more money was invested in value-based testing, testing costs might be decreased by 60%, saving $300 billion annually. Regression testing is expensive and challenging in fast growing and evolving systems. It takes a lot of time and effort and raises software maintenance costs. When a code update is implemented, system testing is usually done immediately, and regression testing can be done at the system, integration, or unit level. Time and resource constraints prevent thorough test coverage during regression testing. Testing teams must decide how much regression testing to perform. TCP, test case selection, test case reduction, and retesting are four methodologies used during regression testing. Figure 1 displays regression testing types.

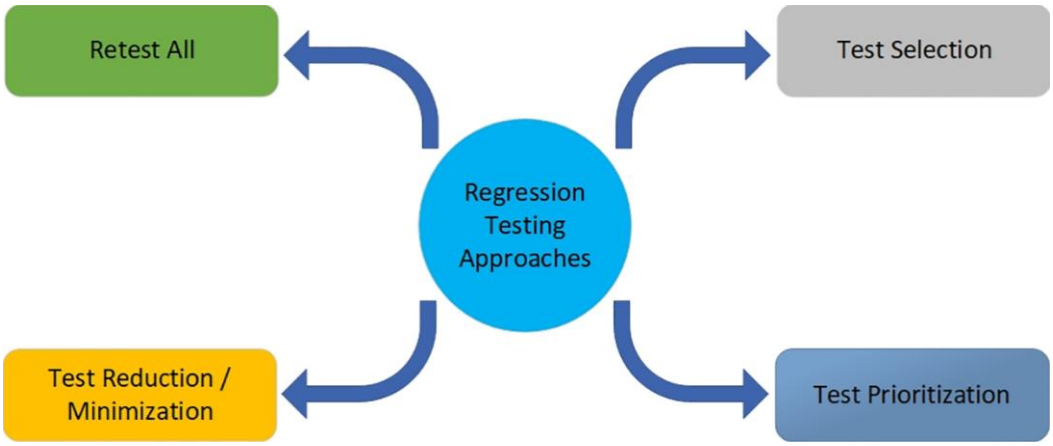

Fig 1. Regression testing approaches.

The strategy of test selection is utilised rather frequently in the industry; yet, given that it is based on selection, there is a possibility of danger associated with it. It is not possible to guarantee that the test case reduction process will result in the removal of only irrelevant test cases from the pool of test cases. “TCP, on the other hand, does not reduce the number of test

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

cases in the test suite nor does it delete any of them". As a consequence of this, it is more dependable, secure, and commonly utilised in actual clinical settings. There is a significant amount of investigation being place "in this field".

"Test case prioritization is one of the ways for optimized regression testing. Rothermel *et al*. defined the TCP problem as follows. *Suppose T is a test suite, PT is a set of permutations of T, and f is a function from PT to real numbers, f*: *PT→R".*
"Prioritization Goal: *To find a* $T^I \in PT$ *that maximizes f".*
TCP techniques consider test case set size, cost, time, effort, efficiency, defect count, and repetitiveness. Most TCP solutions aim to increase problem detection by prioritising test cases to save time and money (APFD). Two TCP alternatives have been suggested. Value-based and neutral apparel are examples. When prioritising test cases for regression testing, consider both cost and mistake severity. The value-neutral approach assumes all bugs are equally serious and that testing costs are the same regardless of complexity. In reality, this premise is rare. Value-based fashion ranks below value-neutral fashion.
The value-neutral TCP approaches assume all errors are handled equally based on severity and cost. "Similar metrics, such as Average Percentage of Statement Coverage (APSC), Average Percentage of Fault Detection (APFD), Total Percentage of Fault Detection (TPFS), Average Percentage of Branch Coverage (APBC), Average Percentage of Function Coverage (APFC), Average Percentage of Condition Coverage (APCC), and Average Percentage of X Elements Coverage (APXEC), have been proposed to measure coverage. All of these measures assume all faults are the same severity, all needs have the same value, and all code statements are relevant; in practise, this is unlikely. Varied needs have different values, and defects vary in severity. Similarly, functions, statements, conditions, branches, and methods can each have a proportionally different value. Most TCP approaches are coverage-based. These unit-level testing methodologies are time-consuming and assume all issues are of similar severity and expense".

BACKGROUND AND RELATED WORK

Test Case Prioritization Problem

In order to improve the effectiveness of the testing process, the test cases are prioritised according to certain criteria in order to discover the greatest number of errors in the shortest amount of time possible. The first recommendation that was made for a full characterisation of the TCP issue was made "by Rothermel. Given three variables: T, an already selected test suite; PT, a collection of all possible prioritizations (orderings) of T; and f, an objective function from PT to the real numbers that, when applied to any such ordering, produces an award value for that ordering. The test suite T has already been selected".
"Problem: Find T′ ∈ PT such that"

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

TCP INITIALIZATION

Classification of Requirement Properties

During the requirements phase, developers might obtain a number of requirement characteristics; they must then implement these using code. During these phases, effective test cases are designed, and each is related to the tested need's features. TCP considers each component of a need since each has its own priority. The relative weight of each requirement attribute is expressed by "Important Value," whose name comes from the term "importance."

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022

We utilise these two criteria to rank the needs' many elements. CP refers to a customer's priority for a required property. Customers will make their decision on this criterion. These levels help us classify the key property requirements. Highest-level qualities relate to system functioning. These features include exception and warning handling. “Higher-level properties handle data input, output, and upgrading”. The remaining attributes level is considered the most basic. Similarly, non-primary needs might be divided into two tiers. Lower-level attributes don't need new data, while higher-level features do. CP IV assigns a level-dependent significance value from 5 to 1 to a needed property r. This value is between those two extremes (r). “Table 1 shows the categorisation and CP IV assignment”.

Developers will apply a developer-assigned priority (DP) to each need characteristic. This priority reflects the complexity of implementing that attribute. Developers make the final decision on this task. The DP has five levels of standards. A needed property r obtains an importance value from 5 to 1, denoted by DP IV, based on the developer's choice (r). Increasing value increases implementation complexity. Higher-complexity required properties likely to have more flaws. Complex implementation makes it harder to catch all problems. Because it has the most problems, its property should be prioritised. Table 2's entire classification can be utilised to determine a person's DP IV designation.

Table 1. Assignment of a “classification and CP IV number”

Classification of CP	Level type	CP_IV
main	system-operation	5
	data-update	4
	non-data-update	3
non-main	data-update	2
	non-data-update	1

Table 2 Assignment and classification according to DP IV

Level type	DP_IV
most-difficult-implementation	5
more-difficult-implementation	4
middle-difficult-implementation	3
simpler-implementation	2
simplest-implementation	1

“According to Table 1 and Table 2, (of a requirement property r can be calculated as following”:

$$IV(r) = \omega_1 * CP\_IV(r) + \omega_2 * DP\_IV(r) \qquad (1)$$

Where (j = 1, 2) represents component importance. Total: $1. If we utilise a large value of, customers will have more say over the required property's priority. If the vital property's priority is high, developers have a stronger duty to obtain it. In the absence of a

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

distinguishing characteristic, each aspect of the experience would be given the same priority, a metric for judging importance. One of the main purposes of learning is to learn how to change one's behaviour for the better.

TCP Initialization

"The relationship matrix can be obtained once the IV of each requirement property is known. Set and test suite of requirements. At least one test case in set T" meets all conditions. Any object may match zero or more attributes. RP (t) = IV(r), hence the problem is solved. The satisfiability relation between T and R is and the algorithm for determining S is.

Algorithm 1 Calculate relationship S

```
1 : For each t_i ∈ T, r_j ∈ R
2 :     if (t_i satisfies r_j ) then
3 :         S(t_i, r_j) = IV(r_j);
4 :     else S(t_i, r_j) = 0;
5 :     endif
6 : endfor
```

As opposed to the 0-1 matrix, which is discussed in, S is the matrix in which the members are actual integers. This sets it apart from the previous matrix. You are able to calculate the priority value of a test case by adding up the individual values (IV) of the attributes that the test case meets. This will give you the test case's priority value.

$$RP(t_i) = \sum_{j=1}^{m} S(t_i, r_j) \qquad (2)$$

"HISTORY-BASED PRIORITIZATION"

"Dynamic adjustment of RP prioritization"

"We first take into account of dynamic adjustment of *IV*. After testing $t \in T$ the flaws that are discovered are first recorded, and then they are assigned to the many properties that t fulfils. This elucidates the total amount of problems that are connected to each essential attribute. If this attempt results in fewer errors than the previous one did, it is considered successful. $IV(r)$ is reduced by their difference, otherwise, $IV(r)$ is added by the difference. Therefore, RP (t) must be recalculated and the test cases must be reordered. Equation 3 denotes the adjustment calculation, where $IV\ (t_i, r_j)$ is the $IV(r_j)$ that satisfies, $IV_n(t_i, r_j)$ is the current value, and $IV_{n-1}(t_i, r_j)$ shows the most up-to-date result of the testing performed on the regressions. The Dvalue fault is the value that is calculated by subtracting the total number of faults that are found in two records that are located next to one another".

$$IV_n(t_i, r_j) = IV_{n-1}(t_i, r_j) + \text{Dvalue\_fault} \qquad (3)$$

RP prioritises. You can choose between "total adjusted RP prioritising and additional adjusted RP prioritisation for dynamic RP adjustment. The total adjusted RP priority is used to rank test cases from highest to lowest RP value. This is done to test the higher-risk scenario sooner in the procedure. If multiple test scenarios have the same highest RP, we'll randomly choose one".

The second technique, Additional modified RP prioritising, dynamically adjusts the "RP of the remaining test cases after each best-case decision. Additional modified RP prioritising is used. Executing the remaining test cases that cover the same attribute as the selected test cases minimises system difficulties. Because the chosen test scenarios cover the property. As a result, we'll define a modified RP prioritising below".

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022

Let Req $(t_i) = \{r|S(t_i, r) > 0\}$, and $|Req(t_i)|$ "be the number of elements in set Req (t_i). After the execution of test case is reduced by $|Req(t_i) \cap Req(t_j)|$. Because it is likely that the RP of the test cases will change after each choice of the best test case, it is essential to select the test case that possesses the highest RP for the session that is now being carried out".

Algorithm 2 Additional adjusted RP prioritization

// assume that the kth is current regression testing.

Input:

$T = \{t_1, t_2, \cdots t_n\}$, $R = \{r_1, r_2, \cdots r_m\}$,

$IV_{k-1}(t_i, r_j)$: IV values of requirement properties satisfied by t_i in last test cycle,

$fault_{k-1}(t_i, r_j)$: number of detected faults in last regression testing,

$fault_{k-2}(t_i, r_j)$: number of detected faults in last two regression testing,

$RP[n]$: set of RP value for each test case,

$Req[n]$: set of requirements satisfied by each test case

Output:

T′: the new test case sequence;

1. **for each** $t_i \in T$, $r_j \in R$
2. $Dvalue\_fault = fault_{k-1}(t_i, r_j) - fault_{k-2}(t_i, r_j)$;
3. $IV_k(t_i, r_j) = IV_{k-1}(t_i, r_j) + Dvalue\_fault$;
4. **endfor**
5. **for each** $t_i \in T, r_j \in R$
6. Calculate $RP(t_i)$ based on the new obtained $IV_k(t_i, r_j)$;
7. **endfor**
8. $T' = \emptyset$;
9. **while** $(T \neq \emptyset)$ **do**
10. t_{best} := SelectBest(T, $RP[n]$);
11. $T' = T' + \{t_{best}\}$;
12. $T = T - \{t_{best}\}$;
13. **for each** $t_i \in T$
14. $RP(t_i)$ = AdditionalStrategy $(Req(t_i), Req(t_{best}), RP(t_i))$;
15. **endfor**
16. **endwhile**

"Algorithm 2 is Additional adjusted RP prioritization method. Lines 1-4 show the method of" IV (t_i, r_j) A change was made after observing a discrepancy in the number of errors discovered during two consecutive rounds of regression testing. We are now able to compute the RP for each test scenario after making the appropriate modifications to the IV. You may find additional adjusted RP prioritisation on lines 8-16, along with its two subsidiary functions, which are called Select Best and Additional Strategy. These functions are located in the same section as the main function. The necessary amount of time to make improvements in Lines 1-4 (t_i, r_j) is O(mn), "and the time required in Line5-7 to recalculate $RP(t_i)$ is O(mn). . Obviously, the time required in Line8-16 is" $O(n^2)$ depends on the amount of time that Algorithm 3 and Algorithm 4 take to complete. As a result, the worst case scenario in terms of temporal complexity is. $O(mn+n^2)$.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

Algorithm 3 SelectBest

Input:
$T = \{t_1, t_2, \cdots t_n\}, RP[n]$
Output:
t_{best}

```
1.  Max_RP = 0;
2.  for each t_i
3.  if (RP(t_i) > Max_RP) then
4.  Max_RP = RP(t_i);
5.  candidateCount = 0;
6.  best = i;
7.  endif
8.  else if (RP(t_i) == Max_RP) then
9.  candidateBest[candidateCount] = i;
10. candidateCount ++;
11. endif
12. endfor
13. if(candidateCount > 0) then
14. best = Random(candidateBest[candidateCount]);
15. endif
```

The third algorithm is called Select Best, and it uses the Greedy Algorithm to choose the best possible answer at each stage of the process. The ranking of priorities is determined by going in declining "order of RP. When the selection process" reaches a point when there are two or highest values that are the same, a single "test case must be chosen at random as the best option. The temporal complexity of O has the highest ranking for Select Best (n)".

Algorithm 4 AdditionalStrategy

Input:
$T = \{t_1, t_2, \cdots t_n\}, Req(t_i), Req(t_j), RP(t_i)$
Output:
$RP(t_i)$

1. **if** (t_i and t_j satisfied the same requirement properties and $t_i \neq t_j$) **then**
2. $RP(t_i) = RP(t_i) - |\boldsymbol{Req(t_i)} \cap \boldsymbol{Req(t_j)}|$;

The sub-function Additional Strategy "in Algorithm 4, which is called by Additional adjusted RP prioritisation but not by Total adjusted RP prioritisation, is what differentiates Additional adjusted RP prioritisation from Total adjusted RP prioritisation". Therefore, the algorithm for Total adjusted RP prioritising won't become too tedious to describe in this context. In accordance with Additional Strategy, it is possible for the "RP values of the remaining test cases to be altered throughout each selection. The O strategy has the highest temporal complexity of the Additional Strategies (1)".

History-based TCP framework

As part of our process, we will supply each test case with its own likelihood, and we will do so by making use of past data. It is not always possible to carry out each and every test case during each and every testing session. This is because of the limitations that are placed on both time and resources. Because of this, we select the hypothetical test situations that have the highest likelihood of occurring. In light of the fact that the equation that we establish for calculating probabilities involves the RP value, we normalise RP as NRP:

$$\mathrm{NRP}(t_i) = \frac{RP(t_i)}{\sum_{j=1}^{n} RP(t_j)}. \qquad (4)$$

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

The probability calculation is defined as follows:

$$\begin{cases} P_1 = NRP_1 \\ P_k = \alpha NRP_k + (1-\alpha)P_{k-1} \quad 0 \le \alpha \le 1\,,\ k \ge 1 \end{cases} \qquad (5)$$

"P_K refers to the k-th probability for each test case executed, and initialization P_1 is the first NRP. The higher the value of P_1 the larger the executing probability of test case t. α is a smoothing constant used to weigh individual history RP values, and the tester can assign it according to the actual circumstance. If α When it comes to selecting the value, if a high value is picked, the probability is primarily dependent on the difference between the most recent two test sessions. Aside from that, the history data of the entire phase is the most crucial thing to take into consideration (beginning with the first session and continuing up to the current session). Figure 1 provides a visual representation of the structure that underpins the method of history-based test case prioritisation".

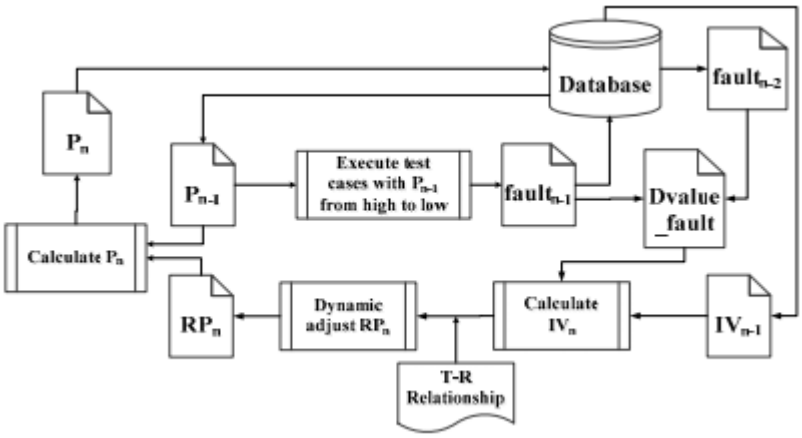

Figure 2. Framework of History-based Test Case Prioritization

In Fig.2 the document icon denotes the input/output document of every execution action. For example P_{n-1} the document icon represents the likelihood of each database-retrieved test case used in the nth iteration of regression testing. Rectangle icon represents an action. Most operations use equations or algorithms. The testing session database is represented as a cylinder. This contains test case ordering and defect counts. This database stores test data. The rounded-corner rectangle shows test scenarios and needed properties. Figure 1 shows data flow. First, when performing regression testing, get all probability-assigned test cases from the database. Next, execute n percent of the most likely test cases. So, we can learn about system flaws. Then, it's computed using the difference between defects in two subsequent records. Dynamic adjustment ensures that RP is calibrated with the T-R connection. Also, the nth relative probability can be calculated. Using Equation 5, each test case's results may be computed and stored as database history. This is a crucial step, not the least.

EMPIRICAL EVALUATION

We do an empirical evaluation in terms of the following research questions in order "to investigate the efficacy of our history-based test case prioritising technique".

RQ1: Whether using our initialization method for the first regression testing improves testing efficiency?

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

It is general information that the "vast majority of research have not focused on first-time ordering" Additionally, the startup of test instances is typically produced at random, as this is the standard practise. The goal of this line of inquiry is to determine whether or not the method of initialization that we use is more efficient than the way of initialization that is based on randomization.

"RQ2: Is our history-based method more effective than other existing test case prioritization techniques?"

"Our second line of inquiry analyses whether or not our history-based method can detect faults earlier than other traditional test case prioritisation strategies, such as random prioritisation and RP-based prioritisation, which are currently being utilised in the industry".

"RQ3: If there is time constraint, whether our history-based method can still improve testing efficiency"

Due to the fact that there is a certain amount of time at our disposal, we are going to work on the assumption that we will only be able to execute n percent (20 percent or 50 percent) of all of the test cases. As part of this line of investigation, one of the questions that we want to answer is whether or not the technique that we use that is based on history may still have a higher rate of identifying flaws.

Prioritization strategies

Following this, in a condensed fashion, we will talk about four distinct test case prioritising approaches for empirical comparison.

TCP initialization is the technique that we propose to employ for the initialization process, and Section III provides a more in-depth discussion of it than was presented in the previous section. To accomplish this goal, it first sorts the test cases in accordance with the importance of the required attributes. We determined that a weight of 0.5 and 0.5, respectively, would be appropriate. When random prioritising is being used, each test case is given an order that is chosen at random. This method of prioritisation is the simplest of the three as it does not include any technical procedures and is the least complicated. Because of the non-parametric character of random approach, we utilised random prioritising on a number of different occasions for each experiment [15]. This was done so that we could compare our results more accurately.

RP-based prioritising organises test cases in the order of decreasing RP (requirement priority) values [3, 14], so that the test case with the highest RP can be executed first if that is the desired outcome. This enables a more effective utilisation of the available testing resources. The methodology that we recommend, known as a history-based prioritisation, is broken down and discussed in greater detail in Section IV of this document. It is able to dynamically change the sequence of the test cases in accordance with the historical fault information after each test cycle has been completed and it has been put through its paces. The smoothing constant has been given the value of 0.8 so that we can account for the change that took place between the most recent two testing sessions.

Evaluation Metrics

APFD

The APFD (also known as the weighted average of percentage of faults discovered) is concerned with the rate of fault detection that takes place during the course of a test suite. Another name for this metric is the weighted average of percentage of faults discovered. The higher the value, the earlier problems can be found while testing a programme; this benefit

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

increases proportionally with the value. The APFD methodology is based on the concept that there is a high degree of similarity between two different orders, namely, faults and costs. The APFD can be determined by applying the formula that is as follows:

$$APFD = 1 - \frac{TF_1 + TF_2 + \cdots + TF_m}{nm} + \frac{1}{2n} \qquad (6)$$

Where n is the total number of test instances and m is the total number of errors that occurred during those test instances. The notation TF refers to the number of the first test case in the execution sequence T that locates fault I, and it stands for "fault identifier."

Fault Detection Rate

Because our experiment required a significant investment of effort on our part, there were times when we were unable to carry out each and every one of the test situations. The ability of a company to detect mistakes within particular intervals of execution time is referred to as the "Fault Detection Rate" (FDR), and it is measured in percentage terms. The following is our interpretation of what that phrase means:

$$FDR = \frac{FDT}{O\_FDT'} \qquad (7)$$

"O FDT is the optimal prioritisation technique that sorts test cases according to the number of faults detected in each case, where FDT is the number of faults detected by the current prioritisation technique during a certain execution time, and where FDT is the number of faults detected by the current prioritisation technique during a certain execution time. This ability to recognise difficulties in the prioritisation process while it is being carried out grows in direct proportion to the value of FDR, which means that the higher the value of FDR the greater the capacity there is to spot faults in the process as it is being carried out".

RESULTS AND ANALYSIS

"In this section, we report the findings of the experiments that we conducted and discuss how those findings relate to the research topics posed earlier".

"RQ1: test efficiency comparison between TCP initialization and random initialization"

The first research topic that we address when we employ our TCP initialization approach for the first round of regression testing is whether or not it is possible to improve the fault-detection capabilities of the system. Experiment 1a, which involved the TCP initialization technique, and Experiment 1b, which involved the random initialization technique, were designed to provide an answer to the question "Will differences in initialization method cause significant differences in fault detection?" Experiment 1a involved the TCP initialization technique, and Experiment 1b involved the random initialization technique. Because we had a preconceived notion that this would turn out to be the case, we decided to construct two experiments to look into the matter. Twenty separate instances of random initialization were carried out so that we could investigate this research issue. This is due to the fact that it does not employ parameters. This response is the mean of twenty different data points, and it is presented for your consideration below. Each process is carried out on the same subject, and the same version of the programme is used for both.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

Table 3 APFD for initialization

Experiment	Initialization	APFD (%)
1a	TCP method	55.48
1b	random method	40.89

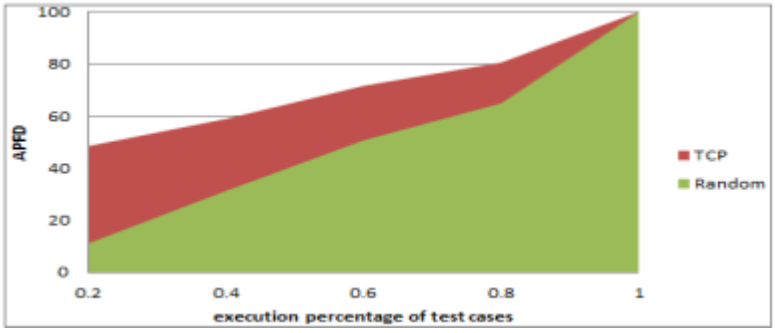

Figure 3.APFD for initialization

Figure 3 examines two approaches' relationships between test cases executed and defects identified. Figure 3 show that our TCP initialization may detect faults faster than random initialization. Table 3 shows that the test sequence's APFD is 55.48% higher than the random initialization. Ordering test cases based on required attributes is more efficient for regression testing than random ordering. Our system's initialization saves time and resources. Although startup is slow, random seems to reduce testing time. This isn't true. Regression testing uses initialization data. If you randomly order test cases when utilising our history-based method to prioritise them, you'll need to calculate that information in subsequent cycles. Global testing equals TCP starting time. Random initialization is preferred to TCP.

"RQ2: test efficiency compared with other existing test case prioritization for regression testing"

Our second research question examines if a history-based prioritising strategy might increase test efficiency relative to existing methods. Experiment2a uses random test case prioritisation, Experiment2b uses RP, and Experiment2c uses history. This research question is answered by three experiments. We use the same experimental approach, but switch from version 2.1 to 2.5 and execute five regression tests. Non-parametric history-based approach is run 20 times per cycle. Other algorithms use it. Figure 3 shows each TCP approach's APFD distribution "for five-times regression testing. The APFD value is vertical, and the number of versions is horizontal. The rhombus legend represents random prioritisation, the square legend RP-based prioritisation, and the circle legend history-based prioritisation. Each legend represents a TCP method. Fig.4 shows that the history-based prioritising technique has higher APFD values than the random strategy in all versions, and that starting with version 2.2, it is also superior to the RP-based method".

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022

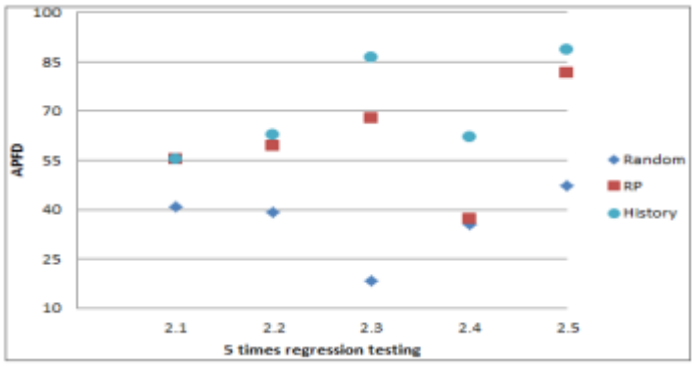

Figure 4.APFD for regression.

"Table 4 shows the regression test results, median, average, and standard deviation" for each TCP technique. Table 4 shows our history-based prioritisation strategy is more efficient than others. History-based prioritising APFD values are greater than random and RP-based prioritising in version 2.1. Version 2.5's history-based prioritisation algorithm achieves 89.54 percent APFD. Higher APFD scores allow for faster problem identification. Due to the unpredictability of the choosing process, random prioritising is generally less effective than average. Unstable. Random prioritisation found the APFD to be lowest. We utilise a one-tail t-test to determine if the measurements' results are useful. We'll assume f1 and f2 are APFD values prioritised using two separate algorithms. Both "are considered":

"H0: f1 = f2, if two techniques have the same effectiveness of fault detection".

"H1: f1> f2, if f1 is significantly better than f2".

If the p-value is less than the significance level ($\alpha = 0.05$), our results prove to have significantly reliability.

Table 5 shows the statistical analysis of five system versions. Table 5 shows history-based prioritisation is better than random and RP. History-based prioritisation has a t statistic larger than 0 and a p-value less than 0.05. History-based prioritisation is significantly different from other methods rather than randomly Table 5's t-value of 3.7267 suggests the variable is approaching null. According to the values, history-based prioritising detects problems more effectively. According to APFD's statistical analyses, t=2.5882 (t>0) and p=0.0304 (p0.05) suggest a substantial difference "between the two procedures. History-based prioritisation is better than role-playing".

Table 5 Statistical analyses of APFD

TCP Techniques	t Stat	p-value (one-tail)
History-based vs Random	3.7267	0.0102
History-based vs RP-based	2.5882	0.0304

"RQ3: time constraint in regression testing our final study question examines whether history-based testing can enhance efficiency when time is limited". We'll only perform n percent of the test scenarios due to time constraints. Experiment3a, 3b, and 3c involved 20% of test instances; Experiment3d, 3e, and 3f involved 50%. We plan six experiments to test our premise that differences in n% will affect fault detection. 20% of test instances were used in Experiments3a, 3b, and 3c. Similar to RQ2's experiments, but with a larger time limit. Table

ISSN PRINT 2319 1775 Online 2320 7876

Research paper © 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022

"6 shows the FDR values of each regression test for each TCP technique, together with their medians, averages, and standard deviations for executing 20% or 50% of the test cases". History-based prioritising has a median value of 78.31% when 20% of test cases are run and an average value of 82.986%, both substantially higher than alternative strategies. History-based prioritisation FDR values reach 100% in the first three runs of regression testing. Fourth-round regression testing decreases the FDR to 78.31%. This is because defects multiply and new ones are added. Random prioritising is the least successful technique since its median and average are lowest. History-based prioritising has the greatest median and average values and the largest FDR in the fifth round of regression testing. History-based prioritisation runs more tests. The number of errors lowers to single digits during the third round of regression testing, and after two rounds of prioritising, we can detect practically all of the faults in 50% of the test cases.

Overall analysis

"History-based prioritising beats random or one-time test case prioritisation (RP-based prioritization). History-based prioritising is excellent for regression testing when running all test cases because it finds flaws quickly. Even with time constraints and not enough time to perform all test cases, history-based prioritising is successful".

Threats to Validity

Threats to internal validity

Effective requirements classification might reduce internal validity. We used a five-point scale to reduce this risk. Weighting provides another risk to the study's internal validity, thus we should base our conclusions on reality. When there is no empirical evidence for weights, using same weights can reduce risk. To lessen the threat, we set and.

Threats to external validity

"We use an industrial project as our experiment objective. The situation of fault occurrence may be different in other projects"

"Table 4: APFD and its median, average, and standard deviation of several TCP techniques (%)"

Experiment	TCP techniques	versions							
		2.1	2.2	2.3	2.4	2.5	Median	Average	Standard Deviation
2a	Random	40.89	39.09	18.24	35.38	47.39	39.09	36.198	9.786
2b	*RP-based*	55.48	56.36	67.65	37.17	81.70	56.36	59.672	14.723
2c	*History-based*	55.48	62.84	86.47	61.94	89.54	62.84	71.254	13.945

"Table 5: FDR and its median, average, and standard deviation of several TCP techniques (%)"

Experiment	*n%*	TCP techniques	versions							
			2.1	2.2	2.3	2.4	2.5	Median	Average	Standard Deviation
3a	20%	Random	18.07	8.330	0	9.090	16.67	9.09	10.432	6.516
3b		RP-based	78.31	41.67	60.00	18.18	50.00	50.00	49.632	19.913
3c		History-based	78.31	63.89	100.0	72.73	100.0	78.31	82.986	14.633
3d	50%	Random	41.88	32.08	0	36.36	16.67	32.08	25.398	15.216
3e		RP-based	76.92	64.15	80.00	63.64	100.0	76.92	76.942	13.279
3f		History-based	76.92	62.26	100.0	72.73	100.0	76.92	82.382	15.157
Number of Faults			102	53	5	11	6			

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

Which we have not made use of thus far. However, the defect occurrence is something that actually happens in industrial projects, and not something that we artificially seed, thus we believe that our experiment can be indicative of a number of different scenarios.

2. CONCLUSION

This study suggests a history-based test case prioritisation and first regression testing startup. Both are regression testing-related. "Our initialization method is more efficient than the random method due to natural faults and non-uniform distribution properties. Our investigations show that our history-based TCP solution has the best fault-detection" capability. Regression testing is affected. Our approach has some flaws. When faults are subdivided by property, redundant faults may result. Future plans may include eliminating unneeded repetition. We want to expand this experiment to incorporate more sophisticated programmes, such as ones with a range of error distributions.

3. REFERENCES

1. Faulk S. R., Harmon R. R., and Raffo D. M. "Value-Based Software Engineering (VBSE)," in Software Product Lines, Donohoe P., Ed. Boston, MA: Springer US, 2000, pp. 205–223.
2. D. Zhang. "Machine Learning in Value-Based Software Test Data Generation," in 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), Nov. 2006, pp. 732–736.
3. Boehm B. W. "Value-Based Software Engineering: Overview and Agenda," in Value-Based Software Engineering, Biffl S., Aurum A., Boehm B., Erdogmus H., and Grünbacher P., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 3–14.
4. Dingsøyr T. and Lassenius C. "Emerging themes in agile software development: Introduction to the special section on continuous value delivery," Inf. Softw. Technol., vol. 77, pp. 56–60, 2016.
5. On the Economics of Requirements-Based Test Case…—Google Scholar." https://scholar.google.com.pk/scholar?hl=en&as_sdt=0%2C5&q=On+the+Economics+of+Requirements-Based+Test+Case+Prioritization&btnG= (accessed Sep. 16, 2018).
6. D. Saff and M. D. Ernst. "Reducing wasted development time via continuous testing," in 14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003., Nov. 2003, pp. 281–292.
7. Zhang X., Onita C. G., and Dhaliwal J. S. "The impact of software testing governance choices," J. Organ. End-User Comput. JOEUC, vol. 26, no. 1, pp. 66–85, 2014.
8. Ramler R., Biffl S., and Grünbacher P. "Value-Based Management of Software Testing," in Value-Based Software Engineering, Biffl S., Aurum A., Boehm B., Erdogmus H., and Grünbacher P., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 225–244.
9. R. Ramler, T. Kopetzky, and W. Platz. "Value-based coverage measurement in requirements-based testing: Lessons learned from an approach implemented in the tosca test suite," in 2012 38th Euromicro Conference on Software Engineering and Advanced Applications, 2012, pp. 363–366.
10. Boehm B. "Value-Based Software Engineering," ACM SIGSOFT, vol. 28, no. 2, p. 12.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper **© 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, S Iss 1,2022**

11. Elbaum, S., Malishevsky, A. G. and Rothermel, G. 2000. Prioritizing test cases for regression testing. In Proceedings of the 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '00). 102-112.
12. Elbaum, S., Malishevsky, A. G., and Rothermel, G. 2002. Test case prioritization: A family of empirical studies. IEEE Trans. Softw. Eng. 28, 2 (Feb. 2002), 159-182.
13. Srikanth, H. and Banerjee, S. 2012. Improving test efficiency through system test prioritization. J. Syst. Softw. 85, 5 (May. 2012), 1176-1187.
14. Kim, J. M. and Porter, A. 2002. A history-based test prioritization technique for regression testing in resource constrained environments. In Proceedings of the 24th International Conference on Software Engineering (ICSE'02). 119-129.
15. Zhang, X., Nie, C., Xu, B., and Qu, B. 2007. Test case prioritization based on varying testing requirement priorities and test case costs. In Proceedings of the 7th International Conference on Quality Software (QSIC'07). 15–24.
16. Huang, Y. C., Peng, K. L., and Huang, C. Y. 2012. A history-based cost-cognizant test case prioritization technique in regression testing. J. Syst. Softw. 85, 3 (Mar. 2012), 626-637.
17. Wang, X. and Zeng, H. 2014. Dynamic test case prioritization based on multi-objective. In Proceedings of the 15th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'14). 1-6.
18. Rothermel, G. and Harrold, M. J. 1996. Analyzing regression test selection techniques. IEEE Trans. Softw. Eng. 22, 8 (Aug. 1996), 529-551.
19. Lin, C. T., Chen, C. D., Tsai, C. S. and Kapfhammer, G. M. 2013. History-based test case prioritization with software version awareness. In Proceedings of the 18th International Conference on Engineering of Complex Computer Systems (ICECCS'13). 171-172.
20. Marijan, D., Gotlieb, A., and Sen, S. 2013. Test case prioritization for continuous regression testing: An industrial case study In Proceedings of the 29th International Conference on Software Maintenance (ICSM'13). 540-543.
21. Yoo, S. and Harman, M. 2012. Regression testing minimization, selection and prioritization: a survey. Softw. Test. Verif. Reliab. 22, 2 (Feb. 2012), 67-120.
22. Saha, R. K, Zhang, L., Khurshid, S., and Perry, D. E. 2015. An Information Retrieval Approach for Regression Test Prioritization Based on Program Changes. In Proceedings of the 37th International Conference on Software Engineering (ICSE'15). 268-279.
23. Arafeen, M. J., and Do, H. 2013. Test case prioritization using requirements-based clustering. In Proceedings of the 6th International Conference on Software Testing, Verification, and Validation (ICST'13). 312-321.
24. Li, S., Bian, N., Chen, Z., You, D., and He, Y. 2010. A simulation study on some search algorithms for regression test case prioritization. In Proceedings of the 10th International Conference on Quality Software (QSIC'10). 72-81.
25. Arcuri, A, and Briand, L. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In Proceedings of the 33rd International Conference on Software Engineering (ICSE'11). 1–10.