Integrated Intelligent Research (IIR)
International Journal of Computing Algorithm
Volume: 05 Issue: 02 December 2016 Page No.72-78
ISSN: 2278-2397

A Novel Prioritized Deciding Factor(PDF) Approach for Directed Acyclic Graph(DAG) Based Test Case Prioritization using Agile Testing Methodology

Alamelu Mangaiyarkarasi[1], M.V.Srinath[2]
[1]Assistant Professor,[2]Director
[1 2]Department of Master of Computer Applications,SengamalaThayaar Educational Trust Women's College, Mannargudi
Email: vsalamu@gmail.com,sri_induja@rediffmail.com

Abstract-Software testing consumes one half of the total developmental resources. Many testers understand that IT management tries to save testing time, by giving them unreasonable deadlines. If a scheduled information date has been established, and dynamic testing can occur only after the code is complete, testers feel pressure of that deadline. This problem faced by many developers and testers till who are all working in the old waterfall model. To overcome the problem of time management of projects, many software industries prefer agile methodology, in agile methodology all works are divided into number of modules. Each module contains decision making, planning, coding, testing, implementation, and so on. Each module follows this process and integrates with the previously completed module. Finally the entire work will be integrated and the time duration of work is shorter than other software development models. But testing occupies the major part of the software development- the maximum time of the allotted time is utilized in testing in agile also. The major objective of this work is prioritization of test cases in Directed Acyclic Graph (DAG) based model. For this purpose introduced a new method named Prioritized Deciding Factor(PDF). It is a value, this value is calculated from other two processes such as Dependency value calculation and K-shingles jaccard similarity and distance value calculation. This paper describes about the prioritization of test cases based on Prioritized Deciding Factor(PDF) value, reduce the execution time of test cases, and increase the fault detection using Average Percentage Fault Detection(APFD) method.

Keyword: Agile Methodology, Prioritized Deciding factor(PDF), K-Shingles Jaccard Similarity, Dependency value, Agile Testing , Time Management, Directed Acyclic Graph(DAG), Average Percentage Fault Detection(APFD)

I.INTRODUCTION

Agile methodology is mainly adopted to increase software development, reduce documentation, and satisfy customers through continuous delivery of the product. Most of the software companies only accept agile based projects because the time duration is smaller than the waterfall model. Waterfall model takes maximum time duration and changes will affect the entire work- from the beginning till the end. However in agile method, changes does not affect at any level. In agile methodology, user interaction is most important at all the level to produce the right product. Customers' requirements are added at any level instantly, and these requirements are created as use cases i.e. user board story; these use cases are split into many test cases. Agile testing involve all members of the cross functional agile team to ensure the business values desired by customer. It includes various methods, such as, Test Driven Development(TDD), Feature Driven Development(FDD), Extreme programming(XP), SCRUM, Crystal Clear (CC), Dynamic Systems Development Method(DSDM),and so on.,[1]. Depending upon the type of the testing, the tester designs the test cases and the collection or set of many test cases is known as Test Suite. Therefore, during the testing phase, to perform the testing the tester can decides which test case is going to test first, then executes the software with those test cases, and then check and verifies the results produced by the executions [2].Here we introduced a new method Prioritized Deciding Factor(PDF). The main aim of this method is prioritize the test cases based the Prioritized Deciding Factor(PDF) value, the Prioritized Deciding Factor(PDF) value calculated from the similarity and dependency values. Through this Prioritized Deciding Factor(PDF) value created test cases are prioritized, i.e. if the Prioritized Deciding Factor(PDF) value is greater than 0 then that value is considered and corresponding test cases are selected for the prioritization process otherwise if the Prioritized Deciding Factor(PDF) value is less than zero, it is not selected and corresponding test cases are considered as a fault node in the Directed Acyclic Graph (DAG) list.

A. Overall process of PDF Approach

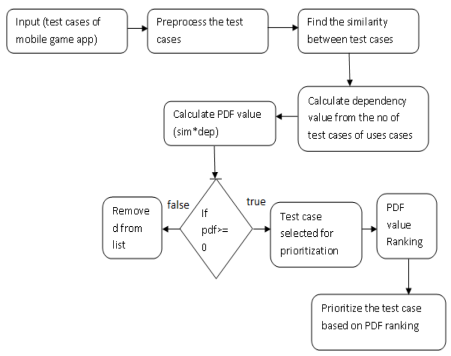

Input (test cases of mobile game app)
Preprocess the test cases
Find the similarity between test cases
Calculate dependency value from the no of test cases of uses cases
Calculate PDF value (sim*dep)
If pdf>= 0
false
true
Remove d from list
Test case selected for prioritization
PDF value Ranking
Prioritize the test case based on PDF ranking

Figure 1: Prioritization Deciding Factor(PDF) Approach

Figure 1 represents the overall process of the Prioritized Deciding Factor(PDF) approach i.e. first of all we can take input from mobile functions as use cases; remove the irrelevant information through preprocessing; split the use cases into number of test cases; find the similarity between the test cases; from the similarity value we can calculate distance value through subtraction from 1; calculate dependency value from use cases; then, calculate Prioritized Deciding Factor(PDF) value from similarity value and dependency value through multiplication. If the Prioritized Deciding Factor(PDF) value

International Journal of Computing Algorithm
Volume: 05 Issue: 02 December 2016 Page No.72-78
ISSN: 2278-2397

is greater than or equal to zero, then those test cases are selected for the process of prioritization, unless the test cases are removed from the list. Finally, the selected test cases are prioritized by ranking of Prioritized Deciding Factor(PDF) value in ascending order.

In Section II, we discuss about preprocessing. The main aim of preprocessing is to remove unnecessary and irrelevant test cases before code writing. In section III, we discuss the process of finding the dependency value between test cases. In section IV, we describe the calculation of similarity value between test cases. In section V, we discuss the calculation of Prioritized Deciding Factor(PDF) value from dependency value and similarity value, and last, we discuss as to how to prioritize the test cases based on the Prioritized Deciding Factor(PDF) value.

II. RELATED WORK

ThillaikarasiMuthusamy et al., suggested a new regression test suite prioritization algorithm that prioritizes the test cases, with the goal of maximizing the number of faults that are likely to be found during the constrained execution [2]. Chunrong Fang et Al., suggested a technique that follows ordered sequences of program entities to improve the effectiveness of test case prioritization. The execution frequency profiles of test cases are collected and transformed into ordered sequences. We propose several novel similarity-based test case prioritization techniques based on the edit distances of ordered sequences [4].

SiripongRoongruangsuwan et al., suggested two new efficient prioritization methods to address the above issues. The first method aims to resolve the problem of many test cases assigned with the same weight values. The second method is developed to effectively prioritize multiple suites. [7]. EktaKhandelwal et al., suggested and reviewed the various techniques for test cases prioritization and the various factors responsible for classification of the techniques. All algorithms have their own field of use [3].Sahil Gupta et al., suggested prioritizing test cases, in order to maximize the number of faults. We introduce two criterions on the basis of which we determine a factor which is used to prioritize test cases. After that, the Average percentage of Fault detection (APFD) metric is evaluated such that it is increased as compared to the non –prioritized order. [8]. Vivekananda ReddyD et Al., suggested that the main objective of the proposed method is to implement an effective software testing process without discarding the provided test cases. [9].

Kavitha R and SureshkumarN suggestedan algorithm to prioritize test cases based on the rate of fault detection and fault impact. The proposed algorithm identifies the severe fault at earlier stage of the testing process and the effectiveness of prioritized test case, and comparison of it with un-prioritized ones with the help of Average percentage of Fault detection (APFD)[10]. Praveen RanjanSrivastava suggested one of the performance goals i.e. the fault detection rate, is a measure of how quickly faults are detected during the testing process and presents a new test case prioritization algorithm, which calculates average faults found per minute. It presents the results illustrating the effectiveness of algorithm with the help of Average percentage of Fault detection (APFD) metric. The main aim is to determine the effectiveness of prioritized and non-prioritized case with the help of Average percentage of Fault detection (APFD) [11].

III. PROPOSED WORK

In this proposed work, there are number of processes that would be performed step by step for the prioritization process. Here, prioritization is the main aim of this proposed work and the processes are as follows:

Step1.Preprocessing
Step2.Similarity value calculation
Step3.Distance value calculation
Step4.Dependency value calculation
Step5.Prioritized Deciding Factor(PDF) Value calculation
Step6.Prioritization of test cases
Step7.Directed Acyclic Graph (DAG) representation

Step1. Preprocessing

The initial process of this proposed work is preprocessing. This method is used to remove irrelevant information from the raw data. Here we can take the inputs from the game applications. It is an unstructured data, i.e raw data. The raw data may contain more irrelevant information. From this raw data, we need to define the structured data, such as use cases. Use cases means user board story, which contains number of test cases. The table shows the list of use cases[14],

Table 1 List of Use Cases

S.No.	Use cases
1.	Check for background music and sound effects# on/off, sound,&,background,music
2. 3.	Check for background music and sound effects#Receive the call and check Check for background music and sound effects#verify if sound effects are in sync with action
4. 5.	Check for background music and sound effects#on/off device sound(native sound0 and check Check for background music and sound effects#check for vibration effect if present
6. 7.	User interface#check in landscape portrait mode User interface#check animation ,movement of character,graphics,zoom in/out (all gueatures)

Table 1 represents use cases about mobile functions such as check for background music and sound effects, user interface and multimedia and so on. These are all considered as use cases. After preprocessing the use cases are split into number of test cases. Then all test cases and use cases are identified by unique numbers, such as 1, 2, and so on.

Table 2 List of Use cases and Test cases with Unique Identification

UID	USE CASE	TID	TEST CASE
1	Check for background music and sound	1	On/off sound and backround music
1	Check for background music and sound	2	Receive the call and check
1	Check for background music and sound	3	Verify if sound effects are in sync with action
1	Check for background music and sound	4	On/off device sound(native sound) and check
1	Check for background music and sound	5	Check for vibration effect if present

The table 2 represents, use case 1 , i.e. check for background music and sound effects is the first use case and it contains 5 test cases, i.e. on/off sound background, receive the call and check, and so on. All the listed test cases are related to use case ‘check for background music and sound effects’ i.e. use case 1.

Step2. Similarity Value Calculation
Similarity is a complex concept which has been widely discussed in the linguistic, philosophical, and information theory communities [12]. Similarity means a process to find the same type of words, sentences, and so on. To calculate similarity value, here we consider two test cases[4]. Each sets are delimited by separator like comma. Now each test cases contains number of words, first test case considered as i and second test case considered as i+1 or j ,i.e., i=1 and j=i+1. Each use case contains n number of test cases; here, use case 1 contain 5 test cases, and the first test case 1 compares with test case 2 i.e. 1→2,1→3 up to 4→5, however, this process is not necessary for reverse process. Here, we use a method to find the similarity between the test cases named jaccard similarity and k-shingles. The formula for this method is

JS (T1, T2) = $\frac{(T1 \cap T2)}{(T1 \cup T2)}$ … (1)

Experiment 1:
Test case 1=on/off, sound,&,background,music
Test case 2=receive, the, call, and, check
The Equation 1 Represented as
Sim(1,2)
$$= \frac{\{\ \}}{\{\frac{on}{off}, sound, \&, background, music, receive, the, call, and, check\}}$$
$$= \frac{0}{10} = 0$$

Experiment 2:
Test case 1=on/off, sound,&,background,music
Test case 3= Verify if sound effects are in sync
Sim(1,3)
$$= \frac{\{sound\}}{\{\frac{on}{off}, sound, \&, background, music, verify, if, effects, are, in, sync\}}$$
$$= \frac{1}{11} = 0.7143$$

Step3.Distance value Calculation
The distance value is used to find the shortest distance between the nodes. The threshold of distance is 1. The distance is calculated from the similarity value i.e. similarity value subtracted by 1. That result value is considered as a distance value. The formula for the distance value calculation is
Distance value=1- Similarity value …..(2)
Experiments:
The Equation 2 follows as
D1= 1-0=1
D2=1-0.077= 0.923
D3=1-0.1=0.9

Table 3 represents calculation of the similarity between test case 1 and test case 2, and similarity between test case 1 and test case 3 and so on. Then the distance value is calculated from the similarity value i.e. similarity value subtracted by 1. Here, the distance value is only used to represent Directed Acyclic Graph (DAG) for the valid test cases, and is not used for prioritization process.

Table 3: Distance Value

Start Node	End Node	Similarity	Distance
1	2	0	1
1	3	0.07143	0.928
1	4	0.09091	0.9
1	5	0	1
2	3	0	1
2	4	0.18182	0.81818
2	5	0	1
3	4	0	1
3	5	0.06667	0.9333
4	5	0	1

Step4.Dependency value Calculation
The dependency value is calculated by the average value of the number of test cases. The purpose of dependency value is to prioritize the test cases based on the use cases. Each use cases contain number of test cases. These test cases are one to one interrelated. The formula for the dependency calculation is
$\sum_{i=1}^{n} UCn = 1/\sum_{i=1}^{n} TCn$*…..(3)*
Where UCn means Number of Use casesTCn means number of Test casesThe Equation 3 represented as follows

Experiment 1:
UID-1=TID1+TID-2+TID-3+TID-4
+TID-5=5
UID-1=$\frac{1}{5}$=0.25

Experiment 2:
UID-2=TID-6+TID-7+TID-8+TID-9+TID-10+TID-11+TID-12+TID-13+TID-14+TID-15+TID-16+TID-17
UID-2=$\frac{1}{12}$=0.0833

Table 4 Dependency Value

UID	UID count	Dependency
1	5	0.20000
2	12	0.08333
3	2	0.50000
4	6	0.16667
5	2	0.50000
6	1	1.00000
7	1	1.00000
8	3	0.33333
9	3	0.33333
10	4	0.25000

Table 4 represents the dependency value of test cases. 1 divided by the number of test cases are considered as the dependency value of each use cases. The first uses case contains 5 test cases, and then the dependency value of use case 1 is 0.2 i.e. and so on.

Step5. Calculation of Prioritized Deciding Factor(PDF)
The prioritized deciding factor (PDF) is a new approach to optimize the prioritization process. In software development lifecycle, testing is an important portion, the testing process occupy half of the allotted time. The main aim of the prioritized deciding factor (PDF) is to reduce the testing time of testers. The algorithm for the prioritized deciding factor

(PDF) approach is ***Algorithm for*** Prioritized Deciding Factor(PDF)

```
Inputs:
Computed Similarity, Distance & Dependency values
Output:
        Prioritized Test Cases

Begin
For each x← $UC_i$
        For all $TC_m$ where $m = 1, 2, \dots N$
        TList← add($TC_i$)
        For each $T_i$ from $TList_n$
        Compute PDF ($T_i$ , $T_j$)
If (PDF ($T_i$, $T_j$) > 0)
Add ($T_i$, $T_j$) →Res ($UC_i$)
End
End For
End For
End For
For each ($T_i$, $T_j$)←List($UC_i$)
Compute $D_{Prio}$ =Pr(PDF ($T_i$, $T_j$))
Result←Produce (Pr ($T_i$, $T_j$))
End for
End
```

The formula of the Prioritized Deciding Factor(PDF) approach is

Prioritized Deciding Factor(PDF) value=similarity value X dependency value . ………..(4)

The algorithm represents the use cases, which are separated from the test cases and added to the TList. Then compute Prioritized Deciding Factor(PDF) value from similarity and dependency values. If Prioritized Deciding Factor(PDF) value is greater than zero, then the test cases are selected to the prioritization list, otherwise it is considered as false node in the Directed Acyclic Graph(DAG) representation and removed from the list. And then the duplicated nodes are removed from the list and provide rank for the selected Prioritized Deciding Factor(PDF) values. The ranking provided is based on the Prioritized Deciding Factor(PDF) value in ascending. After that, the test cases are prioritized based on the Prioritized Deciding Factor(PDF) value.
The Equation 4 represented as

Experiment 1:
Prioritized Deciding Factor(PDF) value=0X0.20000
=0.00000

Experiment 2:
Prioritized Deciding Factor(PDF) value
= 0.07143X0.08333
=0.1429

Table 5 represents the calculation of Prioritized Deciding Factor(PDF) value, the Prioritized Deciding Factor(PDF) value is found from multiplication of similarity value and dependency value, from this new process a set of values are calculated. Here Similarity value of test case 1→ 2 is 0 because no words are match, and the dependency value of use case 1 is 0.2,then the multiplication of 0 *0.20 is 0and so on. According to this process, the test cases are short listed, i.e which Prioritized Deciding Factor(PDF) value is greater than or equal to zero, that value and test cases only selected and prioritized. Here we consider the first node i.e test case 1→2 of Prioritized Deciding Factor(PDF) value is 0, so it is not selected for the prioritization process, likewise, whose value is less than zero that node should be removed from the list of test cases, which is considered as FALSE node.

Table 5: Prioritized Deciding Factor(PDF) value with similarity, dependency

UID	DAG	SIMILARITY VALUE	DEPENDENCY VALUE	PDF VALUE (SIMI*DEP)
1	1-2	0	0.20000	0.00000
1	1-3	0.07143	0.08333	0.01429
1	1-4	0.09091	0.50000	0.01818
1	1-5	0	0.16667	0.00000
1	2-3	0	0.50000	0.00000
1	2-4	0.18182	1.00000	0.03636
1	2-5	0	1.00000	0.00000
1	3-4	0.00000	0.33333	0.00000
1	3-5	0.06667	0.33333	0.01333
1	4-5	0.00000	0.25000	0.00000

Step6. Prioritization of test cases
The test cases are prioritized based on the Prioritized Deciding Factor(PDF) value. If the Prioritized Deciding Factor(PDF) value is >= 0 then the test cases are considered for the prioritization process otherwise it will be removed from the list. First, we need to merge the Prioritized Deciding Factor(PDF) value with valid nodes. Then, prioritize the test cases based on the prioritized deciding factor (PDF) value. In step 6 include the following process.

Duplicate removal
This process is removing the replication of test cases. For example the Prioritized Deciding Factor(PDF) value of 1→3 and 1→4 are 0.0077 and 0.1 respectively. Here 1 occurs two times. Any test cases can occur one time in the list. According to this process, there are 54 test cases that are selected from the full list without duplication.

Ranking the test cases based on PDF value
The selected Prioritized Deciding Factor(PDF) value and corresponding test cases are ordered by ranking based on Prioritized Deciding Factor(PDF) value from the highest to the smallest. In this method, the Prioritized Deciding Factor(PDF) value in not unique, i.e. the same value for more than test cases will create some confusion during the priority process.

Prioritization of test cases based on PDF value
Here, the test cases are ranked based on Prioritized Deciding Factor(PDF) value, but in the prioritization. The same values are avoided up to the next round selection. For these, the repetition should be avoided.

Table 6 represents the prioritized test case based on the Prioritized Deciding Factor(PDF) value, if Prioritized Deciding Factor(PDF) is >=0 then the test case is selected for prioritization, otherwise the test case would be removed from the list of test cases. A simple method is only applied. Here, the Prioritized Deciding Factor(PDF) value is ordered in ascending order. Based on the Prioritized Deciding Factor(PDF) ascending, the test cases are arranged.

Table 6 The prioritized test cases are shown in the table

PDF RANKING	UID	TID	PDF VALUE	USE CASE	T CASE
1	2	17	0.01740	User Interface	Check other objects too (ex -if its a car race-
2	14	75	0.00175	Multiplayer game	login with registered but
3	6	2	0.00267	User Interface	Check in Landscape/Portrait mode
4	2	29	0.00286	User Interface	Test whether one object overlaps with another
5	2	9	0.00308	User Interface	Test whether one object overlaps with another
6	2	10	0.00334	User Interface	Verify if loading indicator is displayed wherever required
7	2	12	0.00471	User Interface	Test for enable and disable images/icons/buttons etc
8	8	8	0.00534	User Interface	There should not be any clipping (cutted background)
9	2	13	0.00572	User Interface	Check for screen title
10	14	79	0.00637	Multiplayer game	Check user statistics graph
11	1	3	0.00715	Check for background	Verify if sound effects are in sync with action
12	1	1	0.00770	Check for background	ON/OFF sound & background music
13	2	15	0.00800	User Interface	Check scrolling
14	3	18	0.01192	Performance	Check the loading time of a game
15	6	45	0.01250	Time Out	Check for time out
16	5	41	0.01318	Score	score calculation
17	1	21	0.01667	Check for background music and sound effects	ON/OFF sound & background music

Step7. Directed Acyclic Graph(DAG) Representation

The Directed Acyclic Graph(DAG) is a graph to represent the distance between the nodes; the link follow from one node to another in forward. Here, the backward process is not necessary in Directed Acyclic Graph(DAG) representation. Here, the test cases are nodes. We can start the process from first test case; however, false nodes are not considered in the list.

Table 7 Use case 1 and Use case 2 valid nodes are shown

UID	DAG	Distance value
1	1-3	0.928
1	1-4	0.9
1	2-4	0.81818
1	3-5	0.93333
2	6-7	0.93750
2	6-13	0.8750
2	6-14	0.92857
2	6-15	0.83333
2	6-17	0.96154

Directed Acyclic Graph(DAG) Representation

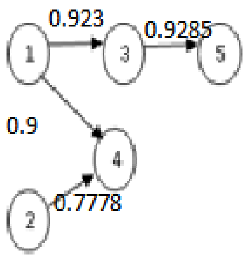
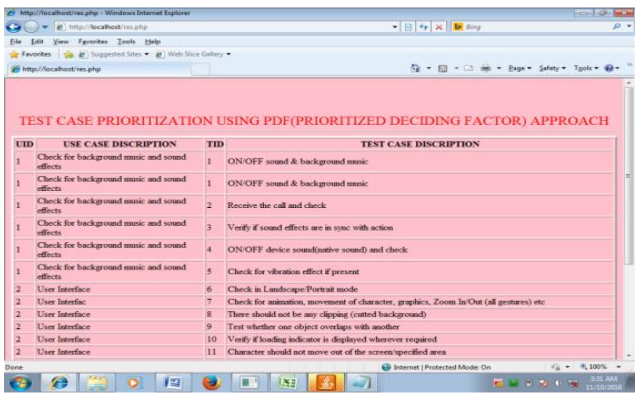


Fig 2. Directed Acyclic Graph(DAG)for use case 1

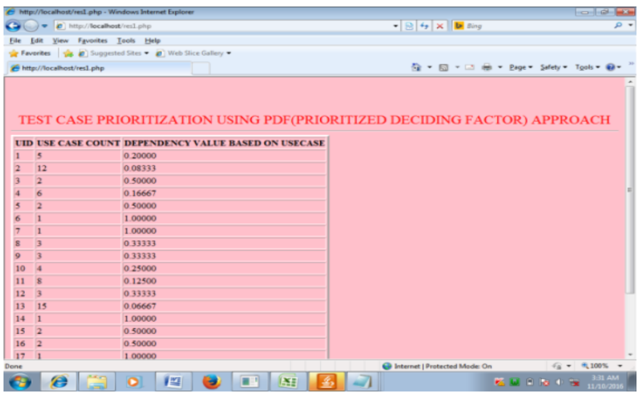
V. RESULT AND DISCUSSION

This section describes t the performance analysis of the prioritized deciding factor (PDF) approach for the following metric,The implementation of the prioritized deciding factor (PDF) is done in the platform of PHP 5.5.12 and MYSQL5.6.17. The sample screens are


http://localhost/res.php - Windows Internet Explorer
File Edit View Favorites Tools Help
Done
Internet | Protected Mode: On

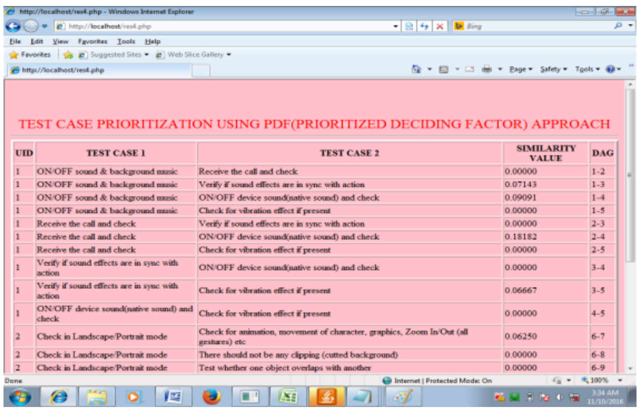
TEST CASE PRIORITIZATION USING PDF(PRIORITIZED DECIDING FACTOR) APPROACH

UID	USE CASE DISCRIPTION	TID	TEST CASE DISCRIPTION
1	Check for background music and sound effects	1	ON/OFF sound & background music
1	Check for background music and sound effects	1	ON/OFF sound & background music
1	Check for background music and sound effects	2	Receive the call and check
1	Check for background music and sound effects	3	Verify if sound effects are in sync with action
1	Check for background music and sound effects	4	ON/OFF device sound(native sound) and check
1	Check for background music and sound effects	5	Check for vibration effect if present
2	User Interface	6	Check in Landscape/Portrait mode
2	User Interfac	7	Check for animation, movement of character, graphics, Zoom In/Out (all gestures) etc
2	User Interface	8	There should not be any clipping (cutted background)
2	User Interface	9	Test whether one object overlaps with another
2	User Interface	10	Verify if loading indicator is displayed wherever required
2	User Interface	11	Character should not move out of the screen/specified area


http://localhost/res1.php - Windows Internet Explorer
File Edit View Favorites Tools Help
Done
Internet | Protected Mode: On

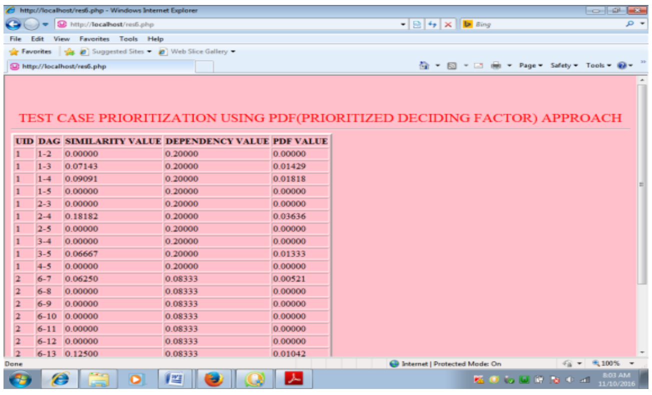
TEST CASE PRIORITIZATION USING PDF(PRIORITIZED DECIDING FACTOR) APPROACH

UID	USE CASE COUNT	DEPENDENCY VALUE BASED ON USECASE
1	5	0.20000
2	12	0.08333
3	2	0.50000
4	6	0.16667
5	2	0.50000
6	1	1.00000
7	1	1.00000
8	3	0.33333
9	3	0.33333
10	4	0.25000
11	8	0.12500
12	3	0.33333
13	15	0.06667
14	1	1.00000
15	2	0.50000
16	2	0.50000
17	1	1.00000


http://localhost/res4.php - Windows Internet Explorer
File Edit View Favorites Tools Help
Done
Internet | Protected Mode: On

TEST CASE PRIORITIZATION USING PDF(PRIORITIZED DECIDING FACTOR) APPROACH

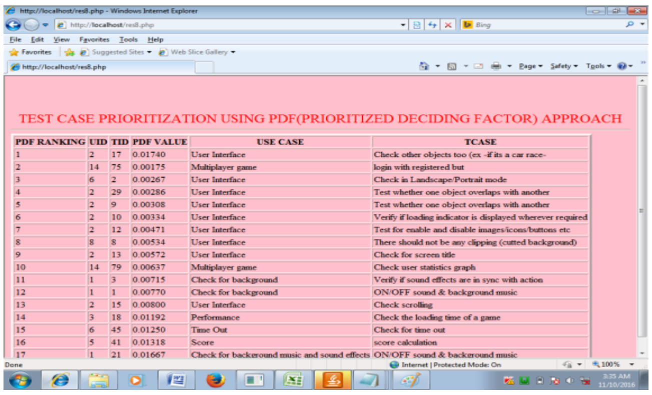
UID	TEST CASE 1	TEST CASE 2	SIMILARITY VALUE	DAG
1	ON/OFF sound & background music	Receive the call and check	0.00000	1-2
1	ON/OFF sound & background music	Verify if sound effects are in sync with action	0.07143	1-3
1	ON/OFF sound & background music	ON/OFF device sound(native sound) and check	0.09091	1-4
1	ON/OFF sound & background music	Check for vibration effect if present	0.00000	1-5
1	Receive the call and check	Verify if sound effects are in sync with action	0.00000	2-3
1	Receive the call and check	ON/OFF device sound(native sound) and check	0.18182	2-4
1	Receive the call and check	Check for vibration effect if present	0.00000	2-5
1	Verify if sound effects are in sync with action	ON/OFF device sound(native sound) and check	0.00000	3-4
1	Verify if sound effects are in sync with action	Check for vibration effect if present	0.06667	3-5
1	ON/OFF device sound(native sound) and check	Check for vibration effect if present	0.00000	4-5
2	Check in Landscape/Portrait mode	Check for animation, movement of character, graphics, Zoom In/Out (all gestures) etc	0.06250	6-7
2	Check in Landscape/Portrait mode	There should not be any clipping (cutted background)	0.00000	6-8
2	Check in Landscape/Portrait mode	Test whether one object overlaps with another	0.00000	6-9



http://localhost/res6.php - Windows Internet Explorer

TEST CASE PRIORITIZATION USING PDF(PRIORITIZED DECIDING FACTOR) APPROACH

UID	DAG	SIMILARITY VALUE	DEPENDENCY VALUE	PDF VALUE
1	1-2	0.00000	0.20000	0.00000
1	1-3	0.07143	0.20000	0.01429
1	1-4	0.09091	0.20000	0.01818
1	1-5	0.00000	0.20000	0.00000
1	2-3	0.00000	0.20000	0.00000
1	2-4	0.18182	0.20000	0.03636
1	2-5	0.00000	0.20000	0.00000
1	3-4	0.00000	0.20000	0.00000
1	3-5	0.06667	0.20000	0.01333
1	4-5	0.00000	0.20000	0.00000
2	6-7	0.06250	0.08333	0.00521
2	6-8	0.00000	0.08333	0.00000
2	6-9	0.00000	0.08333	0.00000
2	6-10	0.00000	0.08333	0.00000
2	6-11	0.00000	0.08333	0.00000
2	6-12	0.00000	0.08333	0.00000
2	6-13	0.12500	0.08333	0.01042



http://localhost/res8.php - Windows Internet Explorer

TEST CASE PRIORITIZATION USING PDF(PRIORITIZED DECIDING FACTOR) APPROACH

PDF RANKING	UID	TID	PDF VALUE	USE CASE	TCASE
1	2	17	0.01740	User Interface	Check other objects too (ex -if its a car race-
2	14	75	0.00175	Multiplayer game	login with registered but
3	6	2	0.00267	User Interface	Check in Landscape/Portrait mode
4	2	29	0.00286	User Interface	Test whether one object overlaps with another
5	2	9	0.00308	User Interface	Test whether one object overlaps with another
6	2	10	0.00334	User Interface	Verify if loading indicator is displayed wherever required
7	2	12	0.00471	User Interface	Test for enable and disable images/icons/buttons etc
8	8	8	0.00534	User Interface	There should not be any clipping (cutted background)
9	2	13	0.00572	User Interface	Check for screen title
10	14	79	0.00637	Multiplayer game	Check user statistics graph
11	1	3	0.00715	Check for background	Verify if sound effects are in sync with action
12	1	1	0.00770	Check for background	ON/OFF sound & background music
13	2	15	0.00800	User Interface	Check scrolling
14	3	18	0.01192	Performance	Check the loading time of a game
15	6	45	0.01250	Time Out	Check for time out
16	5	41	0.01318	Score	score calculation
17	1	21	0.01667	Check for background music and sound effects	ON/OFF sound & background music

Fig.3 Sample screens of proposed work Prioritized Deciding Factor(PDF)

Fault Detection Using Average percentage of Fault detection

Average percentage of Fault detection (APFD) is a standardized metric that is used to find the degree of faults detected. It depends upon the fault criterion considered. Average percentage of Fault detection (APFD) is computed to measure the rate fault detection of coverage based on prioritization techniques. Average percentage of Fault detection (APFD) measures the weighted average of the percentage of faults detected over the life of a test suite. Average percentage of Fault detection (APFD) values ranges from 0 to 100, higher number simply faster faults detection rates [6][10].

Table 8 : shows no of faults and execution time and Prioritized Deciding Factor(PDF) value of test cases

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
F1	*	*	*	*			*			
F2	*				*	*		*		
F3	*	*	*			*		*	*	
F4	*		*				*			*
F5	*		*		*	*	*	*	*	
No of faults	5	2	4	1	2	5	3	3	2	1
PDF value	0.0077	0.02223	0.00715	0.00716	0.00715	0.00267	0.002668	0.005336	0.00308	0.003336

Average percentage of Fault detection (APFD)

$$=1-\frac{TF1+TF2+TF3....+Tfm}{nm}+\frac{1}{2n} \qquad .....(4)$$

Generally test cases to be executed in run time, during the execution there are many faults may occur ,these faults such as syntax error, technical error ,input error, et., to find that error and find the execution time of test cases using Average percentage of Fault detection (APFD) method[11][5][2].

Table 8 represents the faults of each test case and Prioritized Deciding Factor(PDF) values for the Test cases, consider 10 test cases for fault detection, the following table shows the no of faults and the time taken for each test cases

Non prioritized test cases are in order among 80 first 10 test cases only considered
T1,T2,T3,T4,T5,T6,T7,T8,T9,T10
Average percentage of Fault detection (APFD)
= 1-5+1+5+2+4+4+4+3+4+3/10*10+1/2*10
= 0.60
Prioritized test case in the order among 80 test cases, first 10 only considered here
T6,T9,T10 T8,T3,T1,T2,T7,T5,T4
Average percentage of Fault detection (APFD) = 1-5+2+4+1+2+3+3+3+2+1/10*10+1/2*10
= 1-0.27+0.05
= 1-0.32
= 0.68

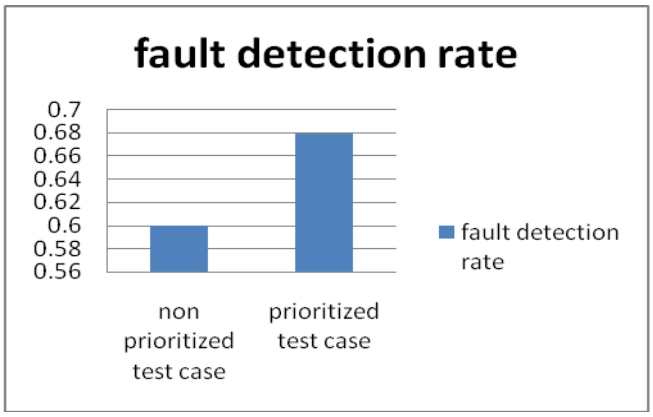


Fig.4 Graph for fault detection rate

Figure 4 represents the fault detection rate in graph, in x axis- percentage of fault detection, and in y axis- test case type i.e. prioritized and non-prioritized. According to the graph, prioritized test case fault detection rate is higher than non prioritized test case fault detection rate.

Execution Time of Test Cases

The execution time of test cases are calculated from the following formula [13]

T.E.T= E.T * NO OF TEST CASES * PERIOD / 8
E.T=3 milli seconds /TC
Number of TC=80
Period of ROI=3 months
=3*80*3/8
=90 ms
=0.09 seconds
Compared with other existing algorithms, this method is better in time execution of test cases. It takes totally 0.09 seconds only to execute 80 test cases.

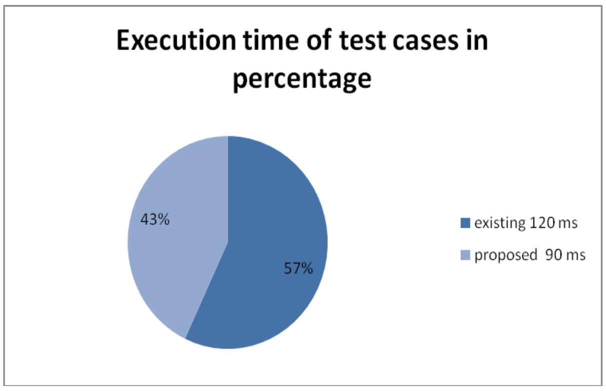


Fig.5.Graph for execution time of test cases

Figure 5 represents the execution time of test cases, Here, existing test cases takes 120 milli seconds for 80 test cases i.e. 57% and proposed 90 milli second is taken for 80 test cases i.e. 43% only, therefore the execution time of this proposed work is lesser than the existing work.

VI. CONCLUSION

This paper proposes an algorithm for test case prioritization using Prioritized Deciding Factor(PDF) method i.e. Prioritized Deciding Factor. The main aim of the proposed work is minimize test cases, reduce the testing time and detect faults in the test cases during execution. Before the calculation of Prioritized Deciding Factor(PDF), we need to compute similarity, distance, and dependency values. The dependency values are computed from use cases. This value is used to make the prioritization process easy. The PDF value is calculated from the similarity and dependency values. The Prioritized Deciding Factor(PDF) based test case prioritization is better than the existing method. To calculate Fault rate, Average percentage of Fault detection (APFD) metric is also applied to detect faults and improve the fault detection rate. The performance analysis of Average percentage of Fault detection (APFD) is better than the existing method. The execution time of test cases is calculated, and those results are also better than the existing method. In future, this work will be implementing in all type of applications. The development of applications are increasing day by day, due to the needs of users. The Prioritized Deciding Factor(PDF) method will help to the users to reduce the time of work and faults will be detected in an easy way.

References

[1] AlameluMangaiyarkarasi V, Dr. Srinath M V and Dr.Omar A Allhayasat. “A Survey on Agile Software Testing Mechanism with Directed Ascyclic Graph (DAG) Based Model in Various Platform,” Australian Journal of Basic and Applied Sciences, Vol. 8, No. 3, pp. 266-273, 2014.

[2] Thillaikarasi M and Seetharaman K, “Test Case Prioritization Techniques on Regression Testing,” International Journal of Innovative Research in Technology & Science (IJIRTS), Volume-2 Issue-6:Published On November 30,2014

[3] EktaKhandelwal and MadhulikaBhadauria, “Various Techniques Used for Prioritization of Test Cases,” International Journal of Scientific and Research Publications, Vol.3, No. 6, ISSN 2250-3153,june 2013,

[4] Fang, C., Chen, Z., Wu, K. et al., “Similarity-Based Test Case Prioritization Using Ordered Sequences of Program Entities.”. Software Qual J (2014) 22: 335. doi:10.1007/s11219-013-9224-0

[5]. Pradeepa R and Vimala Devi K. “Effectiveness of Test Case Prioritization Using AFD Metric: Survey”,International Conference on Research Trends in Computer Technologies (ICRTCT - 2013) Proceedings published in International Journal of Computer Applications® (IJCA) (0975 – 8887)

[6]Anil Mor, “Evaluate the Effectiveness of Test Suite Prioritization Techniques Using APFD Metric,” IOSR Journal of Computer Engineering, Vol. 16, No.4, pp. 47-5, 2014

[7]` SiripongRoongruangsuwan and JirapunDaengdej. “Test Case Prioritization Techniques,” Journal of Theoretical and Applied Information technology.,Vol.18,No.2, 2010.

[8] Sahil Gupta, HimanshiRaperia, EshanKapur, Harshpreet Singh and Aseem Kumar. “A Novel approach for Test Case Prioritization,” International Journal of Computer Science, Engineering and Applications, Vol. 2, No. 3, 2012

[9] Vivekananda Reddy D and Rama Mohan Reddy A. “An Approach for Fault Detection in Software Testing Through Optimized Test Case Prioritization,” International Journal of Applied Engineering Research, Vol. 11, No. 1, pp. 57-63, 2016.

[10] Kavitha R and Sureshkumar N. “Test Case Prioritization for Regression Testing Based on Severity of Fault,” (IJCSE) International Journal on Computer Science and Engineering, Vol. 2, No. 05, pp. 1462-1466, 2010,.

[11] Praveen RanjanSrivastava. “Test Case Prioritization,” Journal of Theoretical And Applied Information Technology,vol. 4 (3), 178-181,2008.

[12] VasileiosHatzivassiloglou, Judith L Klavans, Melissa L. Holcombe, Regina Barzilay, Min-Yen Kan, and Kathleen R. McKeown. “SIMFINDER: A Flexible Clustering Tool for Summarization.” published in in proceedings of the naacl workshop on automatic summarization,2001

[13] http://www.testing-whiz.com/blog/how-to-calculate-roi-for-test

[14] AlameluMangiyarkarasi V, Dr. Srinath M V,An efficient DAGbM-KSJS algorithms for agile software testing,international review on computers and software, ISSN:1828-6003, e-ISSN: 1828-6011.