Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance of Priority Testing in Component-Based Software System

Karambir
Assistant professor
UIET Department of computer Sc. & Engg,
Kurukshetra University Kurukshetra, Haryana, INDIA

Rajni Rani
Research Scholar
UIET Department of Computer Sc. & Engg,
Kurukshetra University Kurukshetra, Haryana, INDIA

Abstract: Scheduling and Prioritizing test cases are one of the most significant tasks in the software testing process. Giving priority to test cases helps you to identify which test cases (high priority) to pick up first while performing priority testing and what test cases (low priority) can be left out. In priority testing test case prioritization will help to improve the rate of fault detection and it provides faster feedback to developers. In this paper, first step conducted a case study on library management system and prioritize component on the basis of risk, cost and impact value and proposed the algorithm that identifies severe fault at the earlier stage of testing process. In next step prioritize test cases in which rate of fault detection, fault impact and test case weightage to be considered and prioritize test cases according to descending order of test case weightage, risk, cost and impact. Effectiveness of non-prioritized test cases and different prioritized test cases measured with the help of APFD metric i.e. average percentage of faults detected per minute. At the end presented the comparisons between non-prioritized test suite and different prioritized test suitess by using APFD metric.

Keywords: Test case prioritization, rate of fault detection (RFD), fault impact (FI), test case weightage (TCW), average percentage of fault detection (APFD), non-prioritized test suite (NPTS), prioritized test suite (PTS)

I. Introduction

Test case prioritization schedule the execution of test cases in a organize manner so that increases the efficiency of software testing. Large numbers of test cases are designed for test software products effectively. But it is not possible to execute every test case due to fix time limitation and some other reasons for instance unavailability of experts. Due to these reasons, there is need of reducing the number of test cases takes during the testing process. Prioritizing test cases helps to reduce the number of test cases. Software is successful only when the Quality of software is increased, cost should be decreased and the product should be according to customer requirement and delivered to the customer in time [1], [2], [3]. In this paper presents an approach to prioritizing the test cases. The proposed prioritization approach prioritizes test cases basis on the risk and cost factors that can occur in a software project. This technique leads to increase the rate of fault detection*:* a measure of how rapidly faults are detected within the given testing process. In this paper described the results of a number of test suites for a given projects and also measured the fault detection rates that achieved by prioritized test suites, compared those rates that achieved by randomly ordered and optimally ordered test suites. The data analysis had shown that proposed test case prioritization in priority testing techniques improve the fault detection rate of test suites. There are some reasons that specified the needs of prioritization methods: 1) prioritization specify which test cases needs to run first, 2) without prioritization waste a lots of time and cost to run the entire test suite. In this paper section 2 and 3 presents the test case prioritization overview and proposed prioritization technique. In section 4 describe an experimentation and analysis and section 5describes a conclusion and future scope.

II. Test Case Prioritization Overview

A. Test case generation

Test case is a combination of inputs, expected outputs and executing functions. We have used JUnit framework [4] for executing the unit tests. JUnit test cases are Java classes it contains test methods that are grouped into test suites.

B. Test case prioritization

Test case prioritization is increase the software testing process efficiency as compared to the execution of test cases in non-prioritized order. Software is test by executing various test cases under different environment. But in case where all test cases are executed for same type of modules changed or unchanged, it results in redundancy. Test case prioritizations schedule test cases those results is increase in the performance of priority testing. Test cases are prioritized in a queue for execution based on priority depending upon different principles. There are two types of prioritization:

1) *Coverage based Test case prioritization ([5],[6],[7],[8]):* There are various prioritization techniques that are implemented based on code coverage information: Stmt_total prioritization, Loop_total prioritization, Branch_total prioritization, Condition_total prioritization.

2) Cost oriented Test case prioritization [9]: In cost oriented test case prioritizations we need to calculate cost of prioritizing each test case. In this section estimate the cost of every test case and then estimate fault severity.

C. Measuring Effectiveness of the various prioritization techniques

For measuring the effectiveness [10] of coverage based prioritization techniques we can use APFD metric.

1) APFD (Average Percentage of Fault Detection [10]: It measures that how quickly faults are found for a given test suites. Higher the range of value shows that faster the rate of fault detection. The APFD values range takes from 0 to 100 and The APFD values represent the area under the curve by plotting percentage of faults detected on y-axis of a graph, and percentage of test suite run on x-axis of a graph. By following the APFD metric formula we can calculate effectiveness of different components of any software. Below given a formula for APFD metric:

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

where, *m* is the the number of exposed faults , *n* is the total number of test cases and *TFi* is the position of first test case in T that exposes fault i.

III. Proposed Prioritization Technique

Case Study

The case study of Library Management System is library management software for controlling and monitoring the daily transaction done in library. This case study gives us information about the book search, book issue, payment bill, track complaints etc. We need to retrieve the details of number of available books and maintain the record of number of new books in the library. We focuses on some fundamental operations in library like add new employee, new books, available books, update new information, search books, view details of employee and student facility to access and return books. It features a familiar and carefully planned, an attractive user interface, combined with good searching, insertion and reporting capabilities.

Activity Diagram: Following defined different activities of library management system:

- User Login and Authentication
- Search Book Operation for Reader
- Acknowledgement and Issue Books to the Users by the Librarian
- Status of Book Updated in Librarian Database.

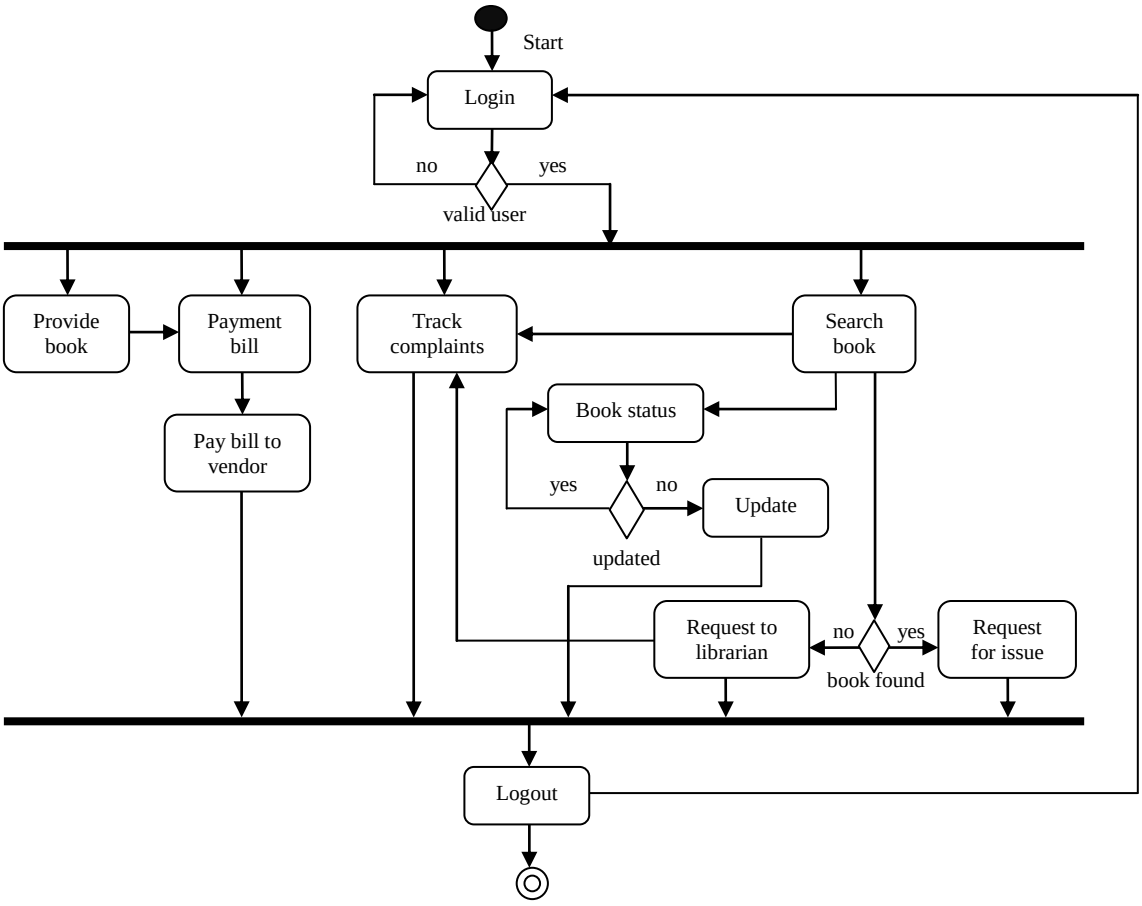


Figure1: Activity Diagram of Library Management System

In this section discusses about prioritization factors that to be considered in test case prioritization. Firstly we will calculate impact factor from randomly assigned value of risk and cost then prioritize the component of library management system according to the descending order of risk, cost and impact; secondly we will take different test cases of a given software projects and calculate rate of fault detection, fault impact and test case weightage and then prioritize test cases according to decreasing order of their test case weightage. For measuring effectiveness of prioritization techniques use APFD metric. At the end presented comparisons between non-prioritized test cases, prioritized test cases, prioritized risk, prioritized cost and prioritized impact by using APFD metric. The factors that to be considered are:

A. Risk

A risk is any unfavourable event that can occur while a project is underway. Risk priority is applied to the software components that specify which component of the software need to reduce risk more intensely than the rest. The component that has high risk, assign them first priority and the component that has low risk assigns them second priority.

B. Cost

It is a constructing or designing cost of a component. Quality of software is increased if cost is decreased so that there is need to minimized cost of component. Minimize cost of component by assigning priority to software componen. If the cost of component is high assign them higher priority and if cost of component is lowest assign them low priority.

C. Impact

Impact of a component measured based on the value of risk and cost. If the value of risk and cost for any component is high than impact value is also high so assign that component first priority, if the value of risk and cost for any component is low than impact value is also low so assign that component second priority. The Impact is calculated from R_c (risk of component) and C_c (cost of component). For calculating impact of component use the following formula:

$$\text{Impact of component } (I_c) = R_c * C_c / 100 \quad (2)$$

D. Rate of fault detection

The average number of faults detected per minute by a test case is called the rate of fault detection. The rate of fault detection of test case j have been calculated by using the number of faults detected divided by time to be taken to find out those faults for each test case of test suite.

$$RFD_{j=} ((\text{number of faults}) / \text{time}) * 10 \quad (3)$$

E. Fault Impact

The testing efficiency can be improved by the focusing on test case that is probable to contain high number of severe faults. So, fault severity value was assigned on the basis of fault impact on the product. Severity value has been assigned based on a 0 to10 point scale as given below:

TABLE 1
Assigning Fault Severity Values

Very High Fault Severity Value	FSV of 10
High Fault Severity Value	FSV of 8
Medium Fault Severity Value	FSV of 6
Less Fault Severity Value	FSV of 4
Least Fault Severity Value	FSV of 2

Below given a formula in which fault impact of $_j$th test case is calculated by severity value of test case j (S_j) divided by high severity value of test case among all the test cases (Max(S)) shown below:

$$FI_j = (S_j / \text{Max}(S)) * 10 \quad (4)$$

F. Test case Weightage

Test case weight of jth test case is computed by adding rate of fault detection of jth test case in fault impact of test case j. Formula of test case weightage is given below:

$$TCW_j = RFD_j + FI_j \quad (5)$$

1.1.1.1 Proposed prioritization algorithm

The proposed prioritization technique is presented in form of the algorithm. The input of algorithm is test suite T, and the output of the algorithm is prioritized test case order. We computed the impact, fault detection rate, fault impact and test case weightage of each test case by using equation ((2), (3), (4) and (5)):

Prioritized factors: Test case weightage, Risk, Cost and Impact
Step 1: **begin**
Step 2: Set PTS_1, PTS_2, PTS_3, and PTS_4 empty
Step 3: **for each** test case t ε NPTS **do**
Step 4: Fault detection rate for each test case ($RFD_j \leftarrow$ (number of faults) / time) *10)
Step 5: Fault impact for each test case ($FI_j \leftarrow (S_j / \text{Max}(S)) * 10$)
Step 6: Test case weightage for each test case ($TCW_j \leftarrow RFD_j + FI_j$)

end for
Step 7: Sort NPTS in descending order on the value of test case weightage and placed in the PTS_1
Step 8: APFD for test suite by following eq. (1)
Step 9: APFD for non-prioritized test suite (NPTS)
Step 10: $APFD_1$ for prioritized test $suite_1$ (PTS_1)
Step 11: **for each** test case t ε NPTS **do**
Step 12: Generate risk value for each component
end for
Step 13: Sort NPTS in descending order on the value of risk and placed in the PTS_2
Step 14: $APFD_2$ for prioritized test $suite_2$ (PTS_2)
Step 15: **for each** test case t ε NPTS **do**
Step 16: Generate cost value for each component
end for
Step 17: Sort NPTS in descending order on the value of cost and placed in the PTS_3
Step 18: $APFD_3$ for prioritized test $suite_3$ (PTS_3)
Step 19: **for each** test case t ε NPTS **do**
Step 20: Generate risk value for each component (Impact of component(Ic)← Rc*Cc/100))
end for
Step 21: Sort NPTS in descending order on the value of Impact and placed in the PTS_4
Step 22: $APFD_4$ for prioritized test $suite_4$ (PTS_4)
Step 23: Compare APFD value for NPTS, PTS1, PTS_2, PTS_3 and PTS_4
Step 24: **end**

IV. Experimentation and Analysis

The experiments were conducted on library management system; in first step we are follow the activity diagram of library management system and assign random values to risk and cost then calculate impact of each component. Prioritize the components according to descending order of risk, cost and impact.

TABLE 2
Non-prioritized component of library management system

Sr. No.	Name of component	Risk(R_c)	Cost(C_c)	Impact(I_c)= R_c * C_c /100
1	Login	40	30	12.0
2	Provide book	27	28	7.56
3	Payment bill	24	37	8.88
4	Track complaints	21	29	6.09
5	Search book	31	38	11.78
6	Pay bill to vendor	25	36	9.0
7	Book status	28	32	8.96
8	Request to librarian	22	31	6.82
9	Request for issue	20	35	7.0
10	Logout	15	25	3.75

TABLE 3
Prioritized component of library management system according to Risk

Sr. No.	Name of component	Risk(R_c)	Cost(C_c)	Impact(I_c)= R_c *C_c/100
1	Login	40	30	12.0
2	Search book	31	38	11.78
3	Book status	28	32	8.96
4	Provide book	27	28	7.56
5	Pay bill to vendor	25	36	9.0
6	Payment bill	24	37	8.88
7	Request to librarian	22	31	6.82
8	Track complaints	21	29	6.09
9	Request for issue	20	35	7.0
10	Logout	15	25	3.75

Prioritized order of risk is: $R_1,R_5,R_7,R_2,R_6,R_3,R_8,R_4,R_9,R_{10}$.

TABLE 4
Prioritized component of library management system according to Cost

Sr. No.	Name of component	Risk(R_c)	Cost(C_c)	Impact(I_c)= R_c * C_c /100
1	Login	40	30	12.0
2	Search book	31	38	11.78
3	Pay bill to vendor	25	36	9.0
4	Book status	28	32	8.96
5	Payment bill	24	37	8.88
6	Provide book	27	28	7.56
7	Request for issue	20	35	7.0
8	Request to librarian	22	31	6.82
9	Track complaints	21	29	6.09
10	Logout	15	25	3.75

Prioritized order of cost is: $C_5,C_3,C_6,C_9,C_7,C_8,C_1,C_4,C_2,C_{10}$.

TABLE 5
Prioritized component of library management system according to Impact

Sr. No.	Name of component	Risk(R_c)	Cost(C_c)	Impact(I_c) = R_c * C_c/100
1	Search book	31	38	11.78
2	Payment bill	24	37	8.88
3	Pay bill to vendor	25	36	9.0
4	Request for issue	20	35	7.0
5	Book status	28	32	8.96
6	Request to librarian	22	31	6.82
7	Login	40	30	12.0
8	Track complaints	21	29	6.09
9	Provide book	27	28	7.56
10	Logout	15	25	3.75

Prioritized order of impact is: $I_1,I_5,I_6,I_7,I_3,I_2,I_9,I_8,I_4,I$

In second step we have different components of software and test these projects using manual testing. We injected 10 faults (F1….F10) varying in severity level in each of the projects. To test the projects, we wrote 10 test cases (T1….T10) for each of the project. We have noted the time taken to find the faults by each test case and severity value.

TABLE 6
Time Taken to Find Out the Fault and the Severity Value

Test cases/ Fault	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
F1						*			*	
F2							*			*
F3	*		*			*			*	
F4		*		*		*		*		
F5								*		*
F6			*		*				*	
F7	*		*		*					
F8										*
F9							*			
F10									*	
Number of Faults	2	1	3	1	2	3	2	2	4	3
Times(ms)	6	9	16	12	8	14	11	13	9	14
Severity Value	4	30	6	4	16	6	24	12	6	12

TABLE 7
Calculate RFD, FI and TCW

T_j	RFD_j	FI_j	TCW_j
T1	3.33	1.33	4.66
T2	1.11	10.0	11.11
T3	1.87	2.0	3.87
T4	0.83	1.33	2.16
T5	2.5	5.33	7.83
T6	2.14	2.0	4.14
T7	1.82	8.0	9.82
T8	1.54	4.0	5.54
T9	4.44	2.0	6.44
T10	2.14	4.0	6.14

Average Percentage of Fault Detection (APFD) can be calculated by following equation (1):

APFD for non- prioritized test suite (NPTS):

$$1 - \frac{6+7+1+2+8+3+1+10+7+9}{10*10} + \frac{1}{2*10} = 0.51$$

APFD = 0.51

Prioritize test case according to descending order of test case weightage. So the prioritized test case order is: T_2, T_7, T_5, T_9, T_{10}, T_8, T_1, T_6, T_3,and T_4.

APFD for prioritized test suite$_1$ (PTS_1):

$$1 - \frac{4+2+4+1+5+3+3+5+2+4}{10*10} + \frac{1}{2*10} = 0.72$$

APFD = 0.72

Prioritize test case according to descending order of risk. So the prioritized test case order is: T_1, T_5, T_7, T_2, T_6, T_3, T_8, T_4, T_9 and T_{10}.

APFD for prioritized test suite$_2$ (PTS_2)

$$1 - \frac{5+3+1+4+7+2+1+10+3+9}{10*10} + \frac{1}{2*10} = 0.60$$

APFD = 0.60

Prioritize test case according to descending order of cost. So the prioritized test case order is: T_5, T_3, T_6, T_9, T_7, T_8, T_1, T_4, T_2 and T_{10}.

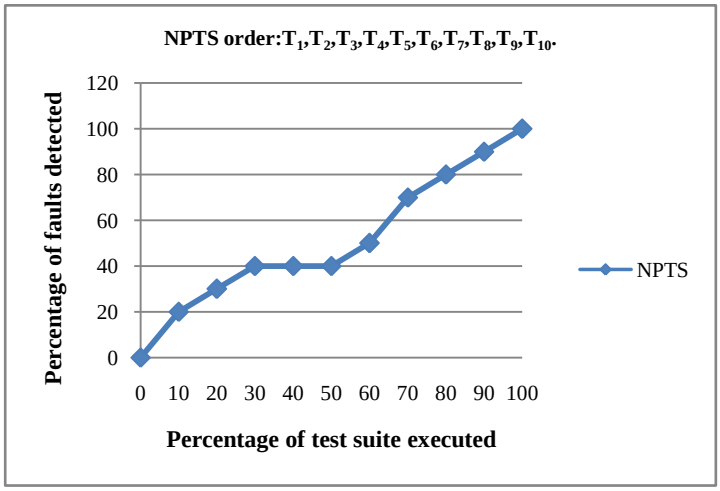
APFD for prioritized test suite$_3$ (PTS_3):

$$1 - \frac{3+5+2+3+6+1+1+10+5+4}{10*10} + \frac{1}{2*10} = 0.65$$

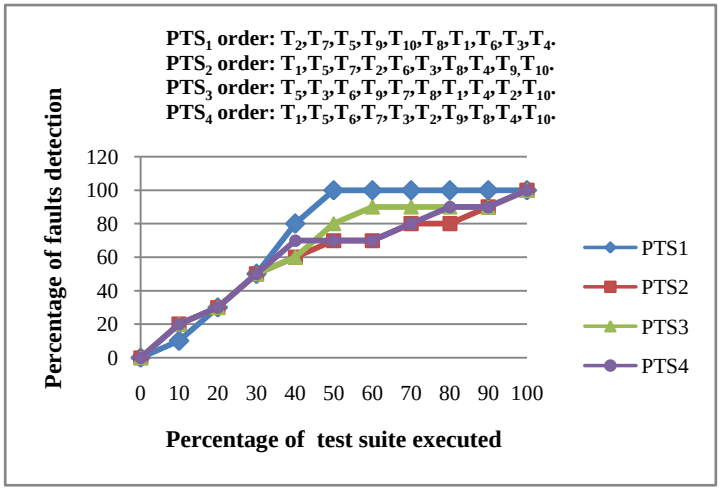
APFD = 0.65

Prioritize test case according to descending order of impact. So the prioritized test case order is: T_1, T_5, T_6, T_7, T_3, T_2, T_9, T_8, T_4 and T_{10}.

APFD for prioritized test suite$_4$ (PTS_4):



Graph 1: APFD Graph for Non-Prioritized Test Suite



Graph 2: APFD Graph for Prioritized Test Suites

$$1-\frac{3+4+1+3+8+2+1+10+4+7}{10*10}+\frac{1}{2*10}=0.62$$

APFD = 0.62

TABLE 8
Time Taken To Find Out the Fault and the Severity Value

Test cases/ Fault	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
F1	*		*			*				
F2					*		*		*	
F3									*	
F4						*				*
F5	*		*		*		*		*	
F6								*		*
F7							*		*	
F8		*		*					*	
F9										*
F10								*		
Number of Faults	2	1	2	1	2	2	3	2	5	3
Times(ms)	11	5	8	6	6	13	10	14	8	6
Severity Value	6	18	8	8	14	6	8	4	6	6

TABLE 9
Calculate RFD, FI and TCW

T_j	RFD_j	FI_j	TCW_j
T1	1.82	3.33	5.15
T2	2.0	10.0	12.0
T3	2.5	4.44	6.94
T4	1.67	4.44	6.11
T5	3.33	7.78	11.11
T6	1.54	3.33	4.87
T7	3.0	4.44	7.44
T8	1.42	2.22	3.64
T9	6.25	3.33	9.58
T10	5.0	3.33	8.33

APFD for non- prioritized test suite (NPTS):

$$1-\frac{1+5+9+6+1+8+7+2+10+8}{10*10}+\frac{1}{2*10}=0.48$$

APFD = 0.48

Prioritize test case according to descending order of test case weightage. So the prioritized test case order is: T_2, T_5, T_9, T_{10}, T_7, T_3, T_4, T_1, T_6, and T_8.

APFD for prioritized test suite$_1$ (PTS_1):

$$1-\frac{6+2+3+4+2+4+3+1+4+10}{10*10}+\frac{1}{2*10}=0.66$$

APFD = 0.66

Prioritize test case according to descending order of risk. So the prioritized test case order is: T_1, T_5, T_7, T_2, T_6, T_3, T_8, T_4, T_9 and T_{10}.

APFD for prioritized test suite$_2$ (PTS_2):

$$1-\frac{1+2+9+5+1+7+3+4+10+7}{10*10}+\frac{1}{2*10}=0.56$$

APFD = 0.56

Prioritize test case according to descending order of cost. So the prioritized test case order is: T_5, T_3, T_6, T_9, T_7, T_8, T_1, T_4, T_2 and T_{10}.

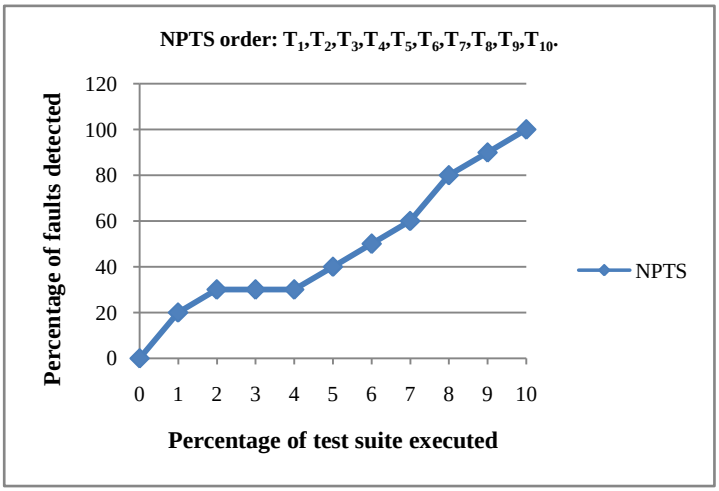
APFD for prioritized test suite$_3$ (PTS_3):

$$1-\frac{2+1+4+3+1+6+4+4+10+6}{10*10}+\frac{1}{2*10}=0.64$$

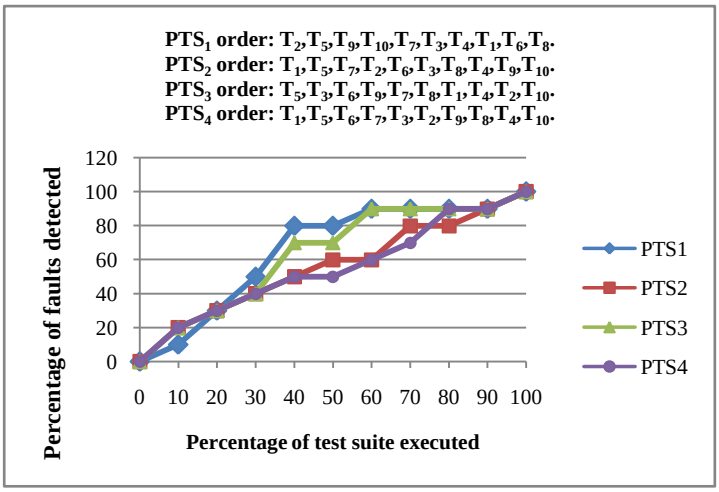
APFD = 0.64

Prioritize test case according to descending order of impact. So the prioritized test case order is: T_1, T_5, T_6, T_7, T_3, T_2, T_9, T_8, T_4 and T_{10}.

APFD for prioritized test suite$_4$ (PTS_4):



Graph 3: APFD Graph for Non- Prioritized Test Suite



Graph 4: APFD Graph for Prioritized Test Suites

$$1-\frac{1+2+7+3+1+8+4+6+10+8}{10*10}+\frac{1}{2*10}=0.55$$

APFD = 0.55

TABLE 10
Time Taken To Find Out the Fault and the Severity Value

Test cases/ Fault	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
F1	*		*			*				
F2		*		*					*	
F3								*		*
F4	*		*		*		*			
F5									*	
F6		*				*				*
F7			*	*			*			
F8	*		*		*					
F9						*				*
F10									*	
Number of Faults	3	2	4	2	2	3	2	1	3	3
Times(ms)	7	9	10	8	12	8	10	12	9	13
Severity Value	8	6	6	10	8	12	14	20	12	8

TABLE 11
Calculate RFD, FI and TCW

T_j	RFD_j	FI_j	TCW_j
T1	4.28	4.0	8.28
T2	2.22	3.0	5.22
T3	4.0	3.0	7.0
T4	2.5	5.0	7.5
T5	1.67	4.0	5.67
T6	3.75	6.0	9.75
T7	2.0	7.0	9.0
T8	0.83	10.0	10.83
T9	3.33	6.0	9.33
T10	2.30	4.0	6.30

APFD for non- prioritized test suite (NPTS):

$$1-\frac{1+2+8+1+9+2+3+1+6+9}{10*10}+\frac{1}{2*10}=0.63$$

APFD = 0.63

Prioritize test case according to descending order of test case weightage. So the prioritized test case order is: T_8, T_6, T_9, T_7, T_1, T_4, T_3, T_{10}, T_5, and T_2.

APFD for prioritized test suite$_1$ (PTS_1):

$$1-\frac{2+3+1+4+3+2+4+5+2+3}{10*10}+\frac{1}{2*10}=0.76$$

APFD = 0.76

Prioritize test case according to descending order of risk. So the prioritized test case order is: T_1, T_5, T_7, T_2, T_6, T_3, T_8, T_4, T_9 and T_{10}.

APFD for prioritized test suite$_2$ (PTS_2):

$$1-\frac{1+4+7+1+9+4+3+1+5+9}{10*10}+\frac{1}{2*10}=0.61$$

APFD = 0.61

Prioritize test case according to descending order of cost. So the prioritized test case order is: T_5, T_3, T_6, T_9, T_7, T_8, T_1, T_4, T_2 and T_{10}.

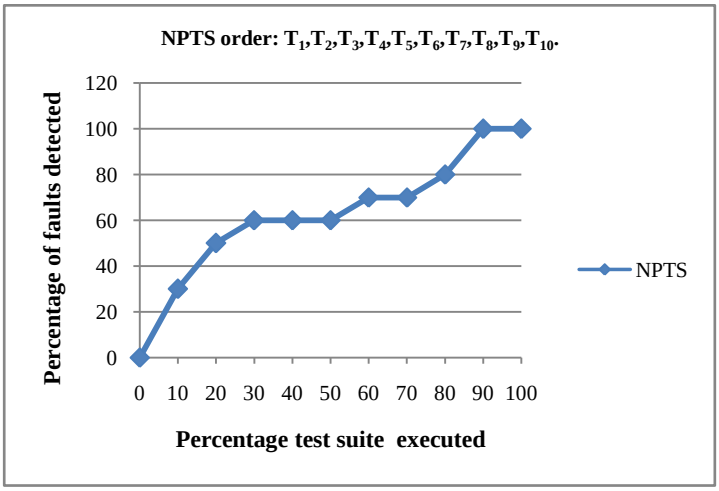
APFD for prioritized test suite$_3$ (PTS_3):

$$1-\frac{2+4+6+1+4+3+2+1+3+4}{10*10}+\frac{1}{2*10}=0.75$$

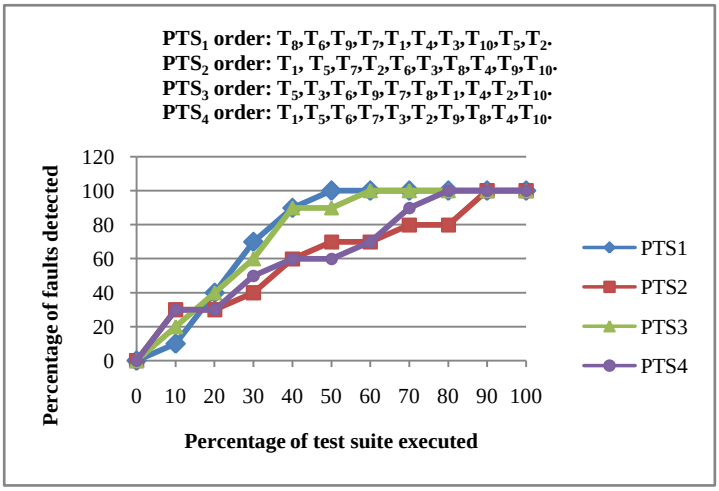
APFD = 0.75

Prioritize test case according to descending order of impact. So the prioritized test case order is: T_1, T_5, T_6, T_7, T_3, T_2, T_9, T_8, T_4 and T_{10}.

APFD for prioritized test suite$_4$ (PTS_4):



Graph 5: APFD Graph for Non-Prioritized Test Suite



Graph 6: APFD Graph for Prioritized Test Suites

$$1-\frac{1+6+8+1+7+3+4+1+3+7}{10*10}+\frac{1}{2*10}=0.64$$

APFD = 0.64

Comparison between prioritized and non- prioritized test case

The comparison between prioritized and non-prioritized test cases shows that number of test cases required to find out faults are less in case of prioritized test cases as compared to non-prioritized test cases. Effectiveness of test cases is measured by using APFD (Average Percentage of Fault Detection) metric that specify how quickly faults are found for a given test suite. APFD value range takes from 0 to 100. Higher the range of value shows that faster the rate of fault detection. Table 12 shows comparisons between non- prioritized test suite and different prioritized test suites by using APFD. It can be observed from table that in all the cases higher the APFD values for prioritized test cases then the non-prioritized test cases, it means the prioritized test case required fewer test cases to find out the faults so that it is more effective than non-prioritized test case.

TABLE 12
Comparisons between Non-Prioritized Test Suite and Different Prioritized Test Suites by Using APFD metric

APFD for non-prioritized test suite	**APFD for Prioritized test suite$_1$**	**APFD for prioritized test suite$_2$**	**APFD for prioritized test suite$_3$**	**APFD for prioritized test suite$_4$**
0.51	0.72	0.60	0.65	0.62
0.48	0.66	0.56	0.64	0.55
0.63	0.76	0.61	0.75	0.64

Conclusion and Future scope

In this paper performed a prioritization technique that improves the effectiveness of prioritization techniques in terms of rate of fault detection for priority testing. The proposed technique considered rate of fault detection, fault impact, test case weightage and prioritization factors (Risk, cost and impact). Proposed algorithm is validated by analyzing different test suites of software. New prioritization technique has a faster rate of fault detection of severe faults as compared to non-prioritized techniques. The proposed technique indicates that the number of test cases required to find out faults are less as compared to randomly ordering test cases technique. Higher the range of value shows that faster the rate of fault detection. Comparisons between non- prioritized test suite and different prioritized test suites by using APFD shows that value for non-prioritized test suite is: 0.51, 0.48 and 0.63, APFD value for prioritized test suite$_1$ is: 0.72, 0.66 and 0.76, APFD value for prioritized test suite$_2$ is: 0.60, 0.56 and 0.61, APFD value for prioritized test suite$_3$ is: 0.65, 0.64 and 0.75 and APFD value for prioritized test suite$_4$ is: 0.62, 0.55 and 0.64. The result shows that in all the cases APFD value for prioritized test suite order is higher than the non- prioritized test suite order, it means prioritized test cases have faster rate of fault detection as compared to non-prioritized test cases, so that the proposed test case prioritization technique is more effective than non-prioritization technique.

In future, we will improve the capability of these techniques by improving the ability to automatically prioritize multiple large test suites with real commercial data. In future work, these areas, we hope to provide software practitioners with cost-effective techniques for improving testing processes through prioritization of test cases.

References

[1] J. Karlsson, K. Rayan , "*A Cost value approach for Prioritizing requirements*," IEEE Software Vol 14, NO 5, 1997.
[2] Maruan Khoury, "*Cost Effective Regression Testing*," October 5, 2006.
[3] Alexy G. Malishevsy, Gregg Rothermel, Sebastian Elbaum, "*Modeling the Cost-Benifts Tradeoffs for Regression Testing Techniques*," Proceeding of the International Conference on Software Maintenance ICSM'02, 2002.
[4] Hyunsook Do., Gregg Rothermel and Alex Kinneer, "*Empirical Studies of Test Case Prioritization in a JUnit Testing Environment*", 15th International Symposium on Software Reliability Engineering(ISSRE), pp.113-124, 2004.
[5] Rothermel, G., R. H. Untch, C. Chu and M.J. Harrold, "*Test case prioritization: An empirical study*", Proceeding of the 15th International Conference on Software Maintenance, Oxford, England, pp. 179-188, 1999.
[6] Rothermel, G., R. H. Untch, C. Chu and M.J. Harrold, "*Prioritizing test cases for regression testing*" ,IEEE transaction Software Engineering, 27: 929-948, 2001a.
[7] Bryce, R.C. and C. Colbourn, "*Prioritized interaction testing for pair-wise coverage with seeding and constraints*", J. Information Software Tech., 48:960-970, 2006.
[8]Leon, D. and A.Podgurski, "*A comparision of coverage-based and distribution-based techniques for filtering and prioritizing test cases*", Proceeding of the 14th International Symposium on Software Reliability Engineering, IEEE Computer Society, Washington, DC,USA., pp. 442-453, 2003.
[9] Elbaum, S.,G. Rothermel, S. Kanduri and A.G. Malishevsky, "*Selecting a cost-effective test case prioritization Techniques*", Software Quality Journal, 12: 185-210, 2004.
[10] H. Do, G. Rothermel, "*On the use of Mutation faults in Empirical Assessments of Test case prioritization Techniques*", IEEE Trans. Software Eng., Vol. 32, No. 9, 2006.

