Article can be accessed online at http://www.publishingindia.com

# A Multi-factor Coverage based Test Case Prioritization Technique for Object Oriented Software

**Vedpal*, Naresh Chauhan****

**Abstract**

Test case prioritization technique creates the sequence of test cases for execution in such a way that the test cases with higher rate of fault detection are executed earlier than those test cases which have lower rate of fault detection. In this paper a new algorithm is proposed to prioritize the test cases based on coverage of object oriented programming factors.The factors are considered on the basis of complexity and probability of errors introduced by them. For the experimental validation and analysis the proposed test case prioritization algorithm is applied on two case studies. The analysed case studies are implemented in C++ language. By using the presented algorithm it helps to reduce the cost and time for testing the software.

**Keywords:** Test Case Prioritization, Object Oriented Testing, Test Case Prioritization of OOS, OOT, TCPOS

## Introduction

Software testing is the crucial part of software development life cycle. Software can't be successfully completed without software testing. Testing the software is the process of validating and verifying of a software program. Thus the main objective of software testing is to maintain and deliver a quality product to client. But software testing consumes a lot of cost in form of time and budget. Also due to time and budget constraints it is not possible to execute all the test cases. To reduce the cost of software testing to some extent, it is necessary to prioritize the test cases, so that more bugs revealing test cases can be executed early. Test case prioritization is important as it increases the efficiency of software testing as compared to the execution of test cases in a non-prioritized order.

The methods that have been previously used for prioritization in object-oriented systems are not adequate. These prioritization techniques basically concern of either functions or class coverage. The other thing is it will not consider other big factor that affect the test case prioritization like naive method which have high effect on testing due to compatibility problem. Similarly there are other factors which have probability to introduce errors in program. In this work, a factors coverage based test case prioritization algorithm is proposed. This algorithm uses nine factors to prioritize test cases. The proposed formula is applied to quantify the factors. Then on the basis of prioritized formula, a prioritized test suite is obtained.

## Related Work

Panigrahi and Mall (2011) proposed a model based approach to prioritize the regression test cases for object-oriented programs. They presented EOSDG (Extended Object Oriented System Dependence Graph) to represent all relevant object-oriented features like inheritance, polymorphism, association and aggregation. For test case prioritization they proposed M-RTP (Model based regression test case prioritization). A forward slice of EOSDG is constructed with respect to each modified model elements to identify model elements that are directly or indirectly affected.

* Department of Computer Engineering, YMCA University of Science & Technology, Faridabad, Haryana, India. Email: ved_ymca@yahoo.co.in

** Department of Computer Engineering, YMCA University of Science & Technology, Faridabad, Haryana, India. Email: nareshchauhan19@gmail.com

Asharf, Rauf and Mahmood (2012) have proposed value based test case prioritization technique. An algorithm which uses six factors: customer priority, changes in requirement, implementation complexity, requirement traceability, execution time, and fault impact of requirement have been used to prioritize the system test cases.

Sultan, Ghani, Baharom and Musa have (2014) presented a regression test case prioritization for object oriented systems based on the dependence graph model of affected program using genetic algorithm. ESDG (Extended System Dependency Graph) was proposed to find the statement level changes in the source code. The identified changes are stored in a file named changed and coverage information for each test case is generated from source code. Then the selected test cases are prioritized using genetic algorithm.

Beena and Sarala (2013) proposed coverage based test case selection and prioritization. They clustered the test cases into three groups outdated, required and surplus. Then by using these clusters test case selection algorithm (TCS) is proposed. Then the output obtained from TCS is given as an input to test case prioritization (TCP).

Shahid and Ibrahim proposed a new code based test case based test case prioritization technique. They proposed a code coverage based test case prioritization algorithm. The test cases which cover the maximum part of code are put at higher priority.

Kavitha and Kumar (2010) proposed a test case prioritization for regression testing based on severity. They proposed a test case prioritization technique based on rate of fault detection and fault impact. This algorithm finds severier fault at the early stage of testing process.

Vedpal, Chauhan and Kumar (2014) proposed a hierarchical test case prioritization technique for OO software. Their proposed an algorithm which works at two levels. At the first level classes are prioritized and at the second level test cases of prioritized classed are ordered.

In this paper a new technique for test case prioritization is presented. The proposed technique works on the basis of the factors. These factors are native method, internal input files,external output files, method, conditional statement, input variable, output variable, line of code, and import packages. For the validation of the proposed technique it has been applied on two case studies. The result shows the efficacy of the proposed technique.

## Proposed Work

The proposed technique prioritizes the test cases by considering nine factors. The test cases are prioritized on the basis of the coverage of the factors. Every factor is assigned a weight which shows the capability of the factor to introduce the error in the software. The capacity of a test case to cover the maximum factors means the test cases covers the maximum code of the software and posses higher ability to detect the maximum errors. The considered nine factors and weight assigns to the factors are as shown in Table 1. The weight is assigned between 0 and 1.

**Table 1: Factors and Weights Assigned**

| *S.No* | *Factor name* | *Weight* |
|---|---|---|
| 1 | Native method | .5 |
| 2 | Internal input files | .2 |
| 3 | External output files | .2 |
| 4 | Import Package | .1 |
| 5 | Conditional statement | .4 |
| 6 | Method | .5 |
| 7 | Input variable | .3 |
| 8 | Output variable | .2 |
| 9 | Line of code | .1 |

After determining the factors covered by the test cases the test cases are ordered on the basis of the test case coverage value (TCCV). Higher the value of the TCCV, higher is the priority of the test case to execute. The TCCV value will be calculated by applying formula 1

$$TCCVi = \sum_{i=1}^{n} cfvalue_{ij} * cfweight_j \quad (1)$$

where cf value is the value of factors covered by the test case and cf weight is the constant weight assigned to the considered factor.

The algorithm of the proposed test case prioritization technique is as shown in algorithm 1

**Table 2: Factors Covered by Test Cases**

| S.No. | Factors | TC1 | TC2 | TC3 | TC4 | TC5 |
|---|---|---|---|---|---|---|
| 1 | Native method | 0 | 0 | 0 | 0 | 0 |
| 2 | Internal input files | 0 | 0 | 0 | 0 | 0 |
| 3 | External output files | 0 | 0 | 0 | 0 | 0 |
| 4 | Import Package | 0 | 0 | 0 | 0 | 0 |
| 5 | Conditional statement | 0 | 1 | 0 | 0 | 0 |
| 6 | Function call | 1 | 1 | 1 | 1 | 1 |
| 7 | Input variable | 3 | 10 | 10 | 2 | 11 |
| 8 | Output variable | 0 | 1 | 1 | 0 | 0 |
| 9 | Line of code | 7 | 19 | 17 | 1 | 1 |

**Table 3: TCCV of Test Cases**

| S.No. | Factors | TC1 | TC2 | TC3 | TC4 | TC5 |
|---|---|---|---|---|---|---|
| 1 | Conditional statement | 0 | 0.4 | 0 | 0 | 0 |
| 2 | Function call | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| 3 | Input variable | 0.6 | 2 | 2 | 0.4 | 2.2 |
| 4 | Output variable | 0 | 0.1 | 0.1 | 0 | 0 |
| 5 | Line of code | 2.1 | 5.7 | 5.1 | 1.8 | 7.5 |
| | TCCV | 3.2 | 8.7 | 7.7 | 2.7 | 10.2 |

## Algorithm 1: Test Case Prioritization

Non-prioritized test suite T

Set T' empty

While ( T!= empty)

Begin

Determine the factors covered by the test case

Calculate the TCCV of test cases by applying the formula

Assign the test cases in decreasing order to T'

end

T' is the set of prioritized test cases.

Let T' be T

## Result and Analysis

For experimental validation and analysis the proposed technique has been applied on the two case studies. The considered case studies are implemented in C++ language. Intentionally errors are introduces in the case studies.

## Case Study 1

In the case study 1 a program in C++ is implemented. This program takes as input the details of student like his name, roll no, marks in different subjects and sport weightage. Then the program finds the total marks of the student, maximum marks in a subject, and average marks of the students and finally displays these various details of the student.

Table 2 shows the factors that covered by the test cases.

After applying the formula 1 the TCCV of the test cases are shown in Table 3

The prioritized orders of the test cases are TC5, TC2, TC3, TC1, TC4.

Table 4 shows the fault detected by the test cases

**Table 4: Fault Detected by Test Case**

| S. No. | TC1 | TC2 | TC3 | TC4 | TC5 |
|---|---|---|---|---|---|
| F1 | * | | | | |
| F2 | | * | * | | * |
| F3 | | * | | | |
| F4 | | * | | | |
| F5 | | | * | | |

| *S. No.* | *TC1* | *TC2* | *TC3* | *TC4* | *TC5* |
|---|---|---|---|---|---|
| F6 | | | * | | |
| F7 | | | | * | |
| F8 | | | | | * |
| F9 | | | | | * |
| F10 | | | | | * |

The effectiveness of the proposed technique is determined by calculating and comparing the APFD value and APFD graph of the prioritized and non-prioritized test suite as shown in Table 5 and Fig. 1.

**Table 5: APFD Value**

| *Prioritization technique* | *APFD (%)* |
|---|---|
| Random Approach | 46% |
| Proposed Approach | 64 % |

## Case Study 2

In the case study 2 a program for different banking operations are implemented in C++. This program maintains the information of a customer like his name, account number and account type. There are different classes for current account and saving account. For depositing amount, withdrawing amount and displaying information there are various functions for different account holders.

Table 6 shows the factors that covered by the test cases.

After applying the formula 1 the TCCV of the test cases are shown in Table 7.

**Fig. 1: Comparison of Proposed and Random Approach**


**Table 6: Factors Covered by Test Cases**

| *S. No.* | | *TC1* | *TC2* | *TC3* | *TC4* | *TC5* | *TC6* | *TC7* | *TC8* |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Native method | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | Internal input files | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | External output files | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | Import Package | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | Conditional statement | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 |
| 6 | Function call | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 |
| 7 | Input variable | 6 | 6 | 5 | 5 | 6 | 6 | 5 | 5 |
| 8 | Output variable | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 9 | Line of code | 36 | 42 | 31 | 36 | 34 | 48 | 31 | 36 |

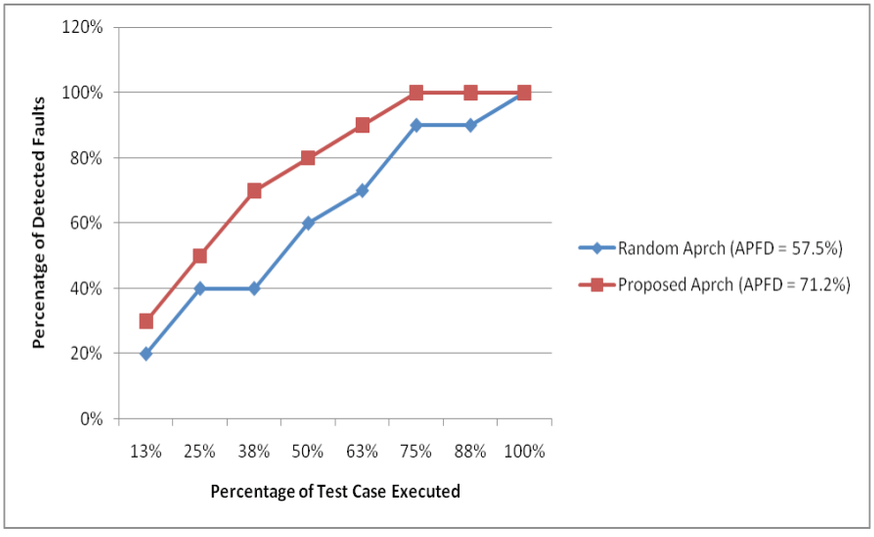
**Table 7: TCCV of Test Cases**

| S.No. | | TC1 | TC2 | TC3 | TC4 | TC5 | TC6 | TC7 | TC8 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Conditional statement | 0 | 0.4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | Function call | 1.0 | 1.0 | 1.0 | 1.5 | 1.0 | 1.0 | 1.0 | 1.5 |
| 3 | Input variable | 1.2 | 1.2 | 1.0 | 1.0 | 1.2 | 1.2 | 1.0 | 1.0 |
| 4 | Output variable | 0.1 | 0.5 | 0 | 0 | 0.5 | 0.5 | 0 | 0 |
| 5 | Line of code | 10.8 | 12.6 | 9.3 | 10.8 | 10.2 | 19.2 | 9.3 | 12.6 |
| | TCCV | 13.1 | 15.7 | 11.3 | 13.3 | 12.9 | 21.9 | 11.3 | 15.1 |

**Table 8: Fault Detected by Test Case**

| | TC1 | TC2 | TC3 | TC4 | TC5 | TC6 | TC7 | TC8 |
|---|---|---|---|---|---|---|---|---|
| F1 | * | * | * | * | * | * | * | * |
| F2 | | | | * | | | | * |
| F3 | | | | | | | | * |
| F4 | | | | | * | | | |
| F5 | | | | | | * | | |
| F6 | | | | | | * | | |
| F7 | | | | * | | | | |
| F8 | * | | | | | | | |
| F9 | | * | | | | | | |
| F10 | | * | | | | | | |

**Fig. 2: Comparison of Proposed and Random Approach**




The prioritized order of the test cases is TC6, TC2, TC8, TC4, TC1, TC5, TC3, and TC7

The faults detected by the test cases are shown in Table 8.

Table 9 and Fig. 2 showthe comparisons of the proposed technique and non-prioritized technique. The analysis shows the effectiveness of the proposed approach.

**Table 9: APFD Value**

| Prioritization technique | APFD (%) |
|---|---|
| Random approach | 57.5% |
| Proposed approach | 71.2% |

## Conclusion

In this paper a new test case prioritization technique for object oriented software is presented that prioritizes test cases based on the coverage of factors by each test case to improve the rate of fault detection in early phase of testing. The result shows that the presented technique enhances the rate of fault detection at early phase in comparison to random order of test cases. For experimental validation and analysis the presented technique is applied on the two case studies that were implemented in C++ language. The result shows the effectiveness of the technique.

## References

Acharya, A. A., Mohapatra, D. P., & Panda, N. (2010). Model based test case prioritization for testing component dependency in CBSD using UML sequence diagram. *International Journal of Advanced Computer Science and Applications*, December, 1(6), 108-113.

Ashraf, E., Rauf, A., & Mahmood, K. (2012).*Value based regression test case prioritization.* Proceedings of World Congress on Engineering and Computer Science, October, 1, 24-26, San Francisco,USA.

Beena, R. & Sarala, S. (2013). Code coverage based test case selection and prioritization. *International Journal of Software Engineering and Applications. November*, 4(6), 39-49.

Chauhan, N. (2010). *Software testing principle and practices*. Oxford University Press.

Kavitha, R., & Kumar, N. S. (2010). Test case prioritization for regression testing based on severity of fault.*International Journal of Computer Science and Engineering*, 2(5), 1462-1466.

Panigrahi, C. R., & Mall, R. (2011). Test case prioritization of object oriented programs. *SETL abs Briefings*, 9(4), 31-40.

Shahid, M., & Ibrahim, S. (2014). A new code based test case prioritization technique. *International Journal of Software Engineering and Its Application*, 8(6), 31-38.

Sultan, A. B. M., Ghani, A. A. A., Baharom, S., & Musa, S. (2014). *An evolutionary regression test case prioritization based on dependence graph and genetic algorithm for object oriented programs.* 2nd International conference on emerging trends in engineering and technology, May, 30-31, London(UK).

Vedpal., Chauhan, N., & Kumar, H. (2014). *A Hierarchical Test Case Prioritization for Object Oriented Software.* International Conference on Contemporary Computing and Informatics(IC3I).