Prioritization of Test Cases scenarios derived from UML Diagrams

Rishi Seth
Department of CSE
Lovely Professional University
Jalandhar-Phagwara, 144 402, INDIA

Sanyam Anand
Department of CSE
Lovely Professional University
Jalandhar-Phagwara, 144 402, INDIA

ABSTRACT
Test Case Prioritization (TCP) is setting priority to each and every test case from test case suit and executes those test cases in descending order. The proposed generic tool will first generate test cases from UML diagram - sequence diagram and prioritize them on basis of following criteria's: prioritization on depending upon depth, prioritization on dependency of number of parameters, prioritization on depending upon code coverage and combination on all the above parameters. The proposed technique will helps in early defect detection, prevents it to move to next phase and thus prevents from software failure. Priority is set according to some criterion and test cases with highest priority are scheduled first. The test case prioritization also minimizes the testing time, increases testing efficiency, reduces the testing cost, earlier identification of high risk defects also provides more time and resource utilization.

Keywords
Test Case Generation (TCG), Test case Prioritization (TCP), Sequence Diagram (SD).

1. INTRODUCTION
Testing is very essential to ensure quality in life cycle of software. As the software size and complexity of software increases, more time as well as effort is needed for testing.
UML helps to design, visualize, construct and document software system [1].
Sequence diagram show dynamic view of software system and is the one of the most important and commonly used in UML diagrams. A sequence diagram shows object interactions arranged in time sequence [2]. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the logical view of the system under development.
We have only dummy skeleton code at design level and not the actual code. So with dummy code, we have functions without logic but with parameters. We have classes with functions and with inherited classes. So on using that stuff we will generate test cases and will prioritize them using above given parameters.
The objective of this work is to develop a tool for test case generation and prioritization technique to detect fault in early stage i.e. at design level so as to improve software quality. Test case Prioritization orders test cases execution based on some priority. Following factors are considered for prioritization: (1) Prioritization on depending upon depth, (2) Prioritization on dependency of number of parameters, (3) prioritization on depending upon code coverage and (4) Combination on all the above parameters. Justification for the selection of these factors is explained in Section III. Rest of the paper is organized as below.
Section II, represents literature survey on Generation and Prioritization of test cases. Section III represents the proposes technique for prioritization. Section IV represents the developed Prioritization Tool. Summary and result of proposed prioritization technique is presented in Section V, VI, VII and VIII. Conclusion and future work is represented in Section IX.

2. RELATED WORK
This section discusses literature survey and overview of work done in background of test case prioritization.
In [3], R. Kavitha and Dr. N. Suresh kumar suggests prioritization of test cases by using various factors such as (a) Customer assigned priority of requirements, (b) Developer perceived code implementation complexity, (c) Changes in requirements, (d) Fault impact. All these factors were implemented in coding phase i.e. in the middle or in last stage but our technique works at initial stage i.e at designing phase. In [4, 11], S. Elbaum, A. Malishevsky and G. Rothermel has experimentally tested that test case prioritization is beneficial for reducing time as well as cost that is put on testing. We have taken UML sequence diagram as it allows complex scenarios to be represented in single diagram. In previous proposed work for deriving test cases [5, 6], the sequence diagram was converted into some other format that was bit complex. Sangeeta Sabharwal, Ritu Sibal and Chayanika Sharma proposes a technique for prioritization of test case scenarios derived from UML activity diagram and state chart diagram using the concept of IF and Genetic Algorithm to identify test paths [2]. G. Rotherrmel and M. Harrold proposed test case prioritization technique based on code coverage and gives priority to those test cases that covers more statements in coding phase [7] but our proposed technique covers code coverage, depth as well as number of parameter. Praveen Ranjan Srivastava proposes test case prioritization technique by calculating (APFD) Average Percentage of Fault Detected with goal of increasing a subset of test suits rate of fault detection [8]. M. Khandai, Arup Acharya and Durga Prasad Mohapatra generate test cases by converting sequence diagram into concurrent composite Graph and then traversing using BFS, DFS and message sequence path criteria technique [9] . Sujata, Mohit Kumar and Dr. Varun Kumar proposes test case prioritization technique that is completely based on requirements of the system and is useful when source code or binary code in unavailable and is only useful in black box testing [10]. Arita Bandyopadhyay and Sudipto Ghosh proposes approach that combines information from UML

sequence models and state machine models. They use state machines to generate multiple execution paths from a message path by analyzing the effect of the messages on state transitions of the system [12]. M. Prasanna and K.R. Chandran proposes a new model based approach for automated generation of test cases by analyzing the dynamic behavior of the objects using Genetic Algorithm's tree crossover to bring out all possible test cases [13].

3. PROPOSED WORK

The proposed idea will generate cases (test cases) from UML sequence diagram in earlier phase i.e. in designing phase. In life cycle of software, firstly the design comes which further is converted into coding phase. If the test cases will be generated in designing phase and if any error is seen during the design of software, it can be corrected easily before coding, so error will be prevented to occur in next phase and will prevent it from increasing that will save our time and cost on fixing it. The most of emphasis should be put on prioritization part so as to make our software good and effective.

Once test cases are generated prioritize them on basis of some following factors:

1. Prioritization on depending upon the depth – if more number of objects are covered, higher the priority level is set.

2. Prioritization on depending upon the number of parameters – if more number of parameters are involved, higher the priority level is set.

3. Prioritization on depending upon code coverage – In some situations most of test cases are involved (e.g. In case of ATM systems much of test cases are involved such as card checking, password checking, balance checking etc. – in such situation covers most of the code) so more will be code coverage - high priority level is set [7].

4. All the above three factors are combined in this factor.

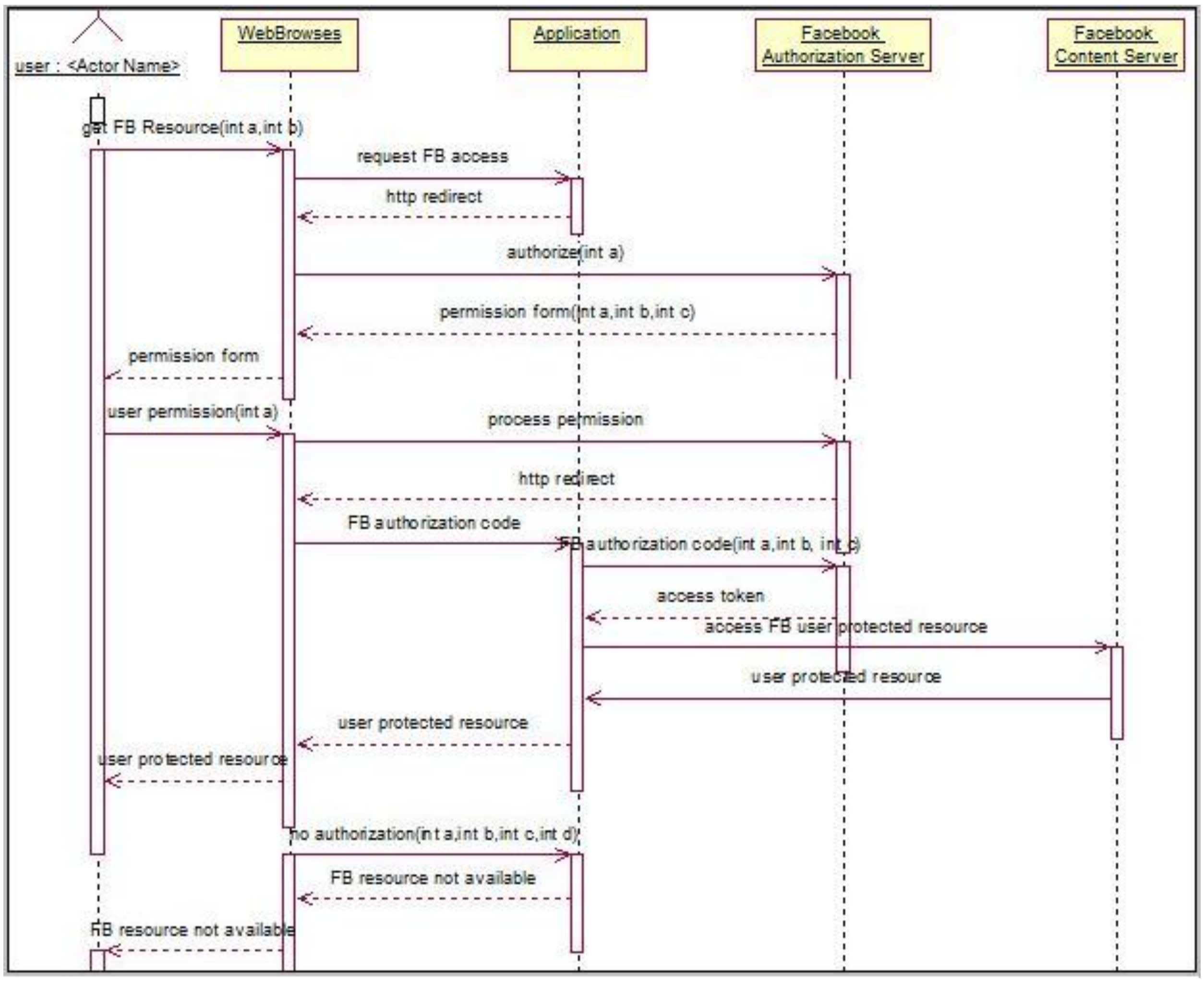
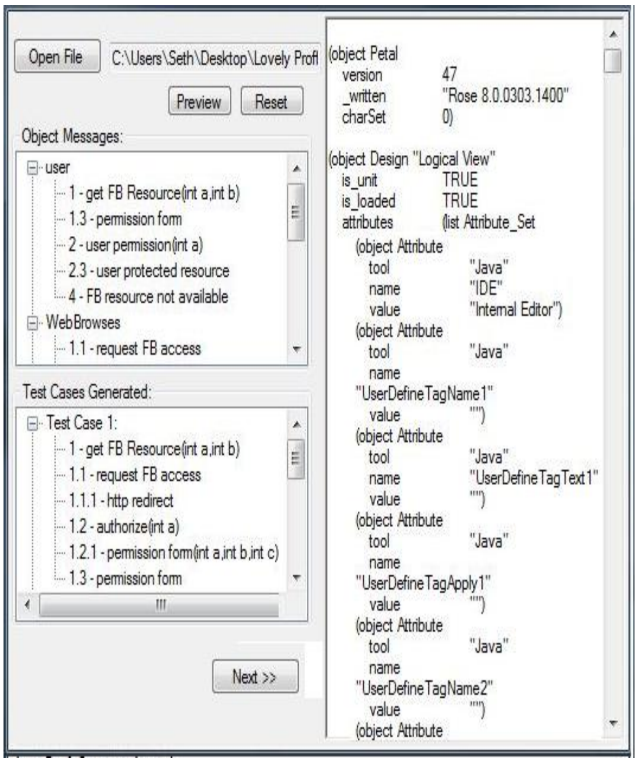


Fig 1: Sequence diagram of FB Authorization in Rational Rose

At design level we only have skeleton code with parameters but without functions. So on using that stuff we will generate test cases and will prioritize them using above given parameters.

4. NEW PRIORITIZATION TOOL

To support automated test case generation and prioritization method, a generic tool is developed in visual studio 2010 using C#, that will generate test cases from UML Sequence diagram that is generated from rational rose .mdl file and will prioritize it according to given parameters.



Open File C:\Users\Seth\Desktop\Lovely Profi

Preview Reset

Object Messages:

- user
  - 1 - get FB Resource(int a,int b)
  - 1.3 - permission form
  - 2 - user permission(int a)
  - 2.3 - user protected resource
  - 4 - FB resource not available
- WebBrowses
  - 1.1 - request FB access

Test Cases Generated:

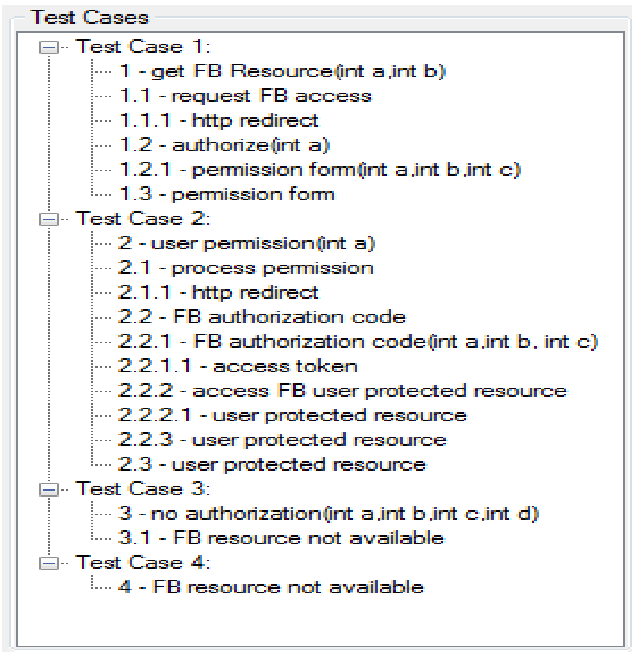
- Test Case 1:
  - 1 - get FB Resource(int a,int b)
  - 1.1 - request FB access
  - 1.1.1 - http redirect
  - 1.2 - authorize(int a)
  - 1.2.1 - permission form(int a,int b,int c)
  - 1.3 - permission form

Next >>

```
(object Petal
  version        47
  _written       "Rose 8.0.0303.1400"
  charSet        0)

(object Design "Logical View"
  is_unit        TRUE
  is_loaded      TRUE
  attributes     (list Attribute_Set
    (object Attribute
      tool           "Java"
      name           "IDE"
      value          "Internal Editor")
    (object Attribute
      tool           "Java"
      name
"UserDefineTagName1"
      value          "")
    (object Attribute
      tool           "Java"
      name           "UserDefineTagText1"
      value          "")
    (object Attribute
      tool           "Java"
      name
"UserDefineTagApply1"
      value          "")
    (object Attribute
      tool           "Java"
      name
"UserDefineTagName2"
      value          "")
    (object Attribute
```

Fig 2: Object Messages and Test case Generation



Test Cases

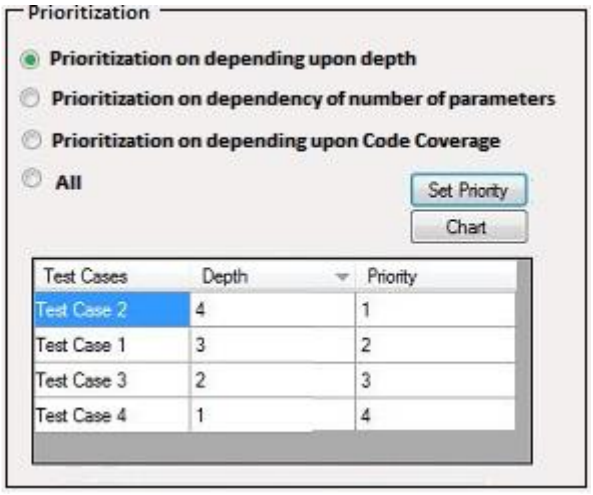
- Test Case 1:
  - 1 - get FB Resource(int a,int b)
  - 1.1 - request FB access
  - 1.1.1 - http redirect
  - 1.2 - authorize(int a)
  - 1.2.1 - permission form(int a,int b,int c)
  - 1.3 - permission form
- Test Case 2:
  - 2 - user permission(int a)
  - 2.1 - process permission
  - 2.1.1 - http redirect
  - 2.2 - FB authorization code
  - 2.2.1 - FB authorization code(int a,int b, int c)
  - 2.2.1.1 - access token
  - 2.2.2 - access FB user protected resource
  - 2.2.2.1 - user protected resource
  - 2.2.3 - user protected resource
  - 2.3 - user protected resource
- Test Case 3:
  - 3 - no authorization(int a,int b,int c,int d)
  - 3.1 - FB resource not available
- Test Case 4:
  - 4 - FB resource not available

Fig 2.1: Generated - Test Cases

The .mdl file is shown in right side of Fig. 2, and left side of Fig. 2 represents object messages. Fig 2.1 shows the Test Cases, that are generated by tracing some patterns from .mdl file.

4.1 Prioritization Depending upon Depth

Fig. 3 represents prioritization depending upon the depth. If more the number of objects are involved in the test case then more important that test case will be and more the priority level it will have. Table in Fig. 3 represents the priority of test case with respect to depth and it is represent graphically in Fig. 3.1.



Prioritization

[✓ Prioritization on depending upon depth]
Prioritization on dependency of number of parameters
Prioritization on depending upon Code Coverage
All

Set Priority
Chart

Test Cases	Depth	Priority
Test Case 2	4	1
Test Case 1	3	2
Test Case 3	2	3
Test Case 4	1	4

Fig 3: Prioritization on depending upon depth

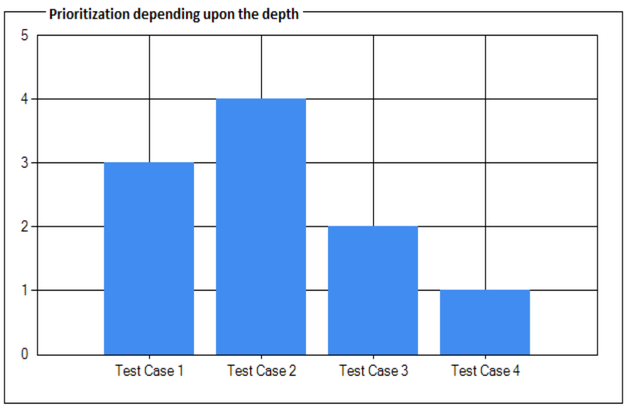
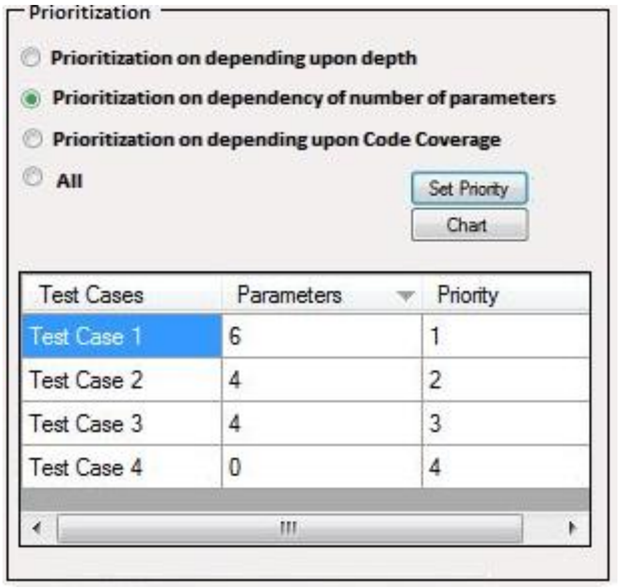


Fig 3.1: Results depending upon depth

4.2 Prioritization on Dependency of Number of Parameters

If the test case contains various function and functions consists of number of parameters. If the function consists of more number of parameters, more important that test case will be in the from the test case suit. Fig. 4 represents the prioritization on dependency of number of parameters and it is represented graphically in Fig. 4.1.



Prioritization on depending upon depth [✓ Prioritization on dependency of number of parameters] Prioritization on depending upon Code Coverage All

Test Cases	Parameters	Priority
Test Case 1	6	1
Test Case 2	4	2
Test Case 3	4	3
Test Case 4	0	4

Fig 4: Prioritization dependency on number of parameters

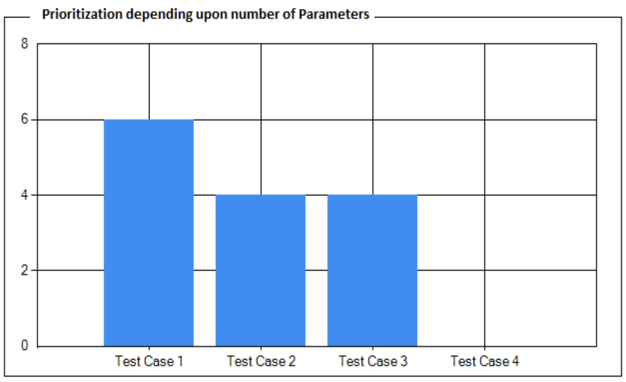
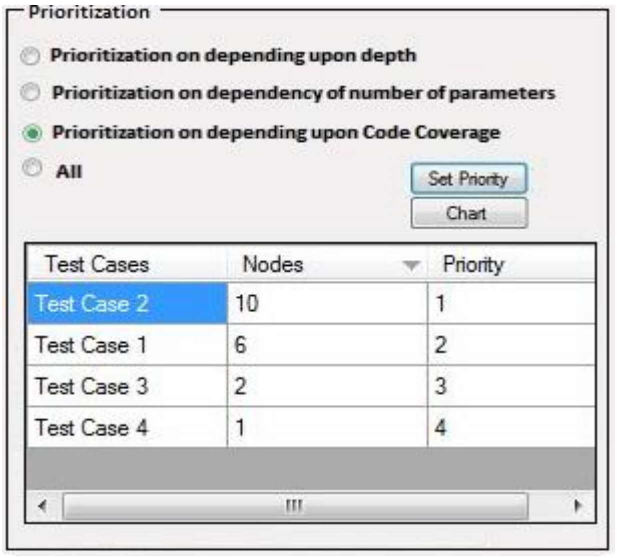


Fig 4.1 Results depending upon no of parameters

4.3 Prioritization on Depending Upon Code Coverage

Code coverage is an important parameter in prioritization of test cases. Code coverage represents the amount of code the particular test case covers. More the number of nodes the test case will have more will be the priority level [7]. Table in Fig. 5 represents the test cases, nodes and priority level and Fig. 5.1 graphical represents the priority level.



Prioritization on depending upon depth Prioritization on dependency of number of parameters [✓ Prioritization on depending upon Code Coverage] All

Test Cases	Nodes	Priority
Test Case 2	10	1
Test Case 1	6	2
Test Case 3	2	3
Test Case 4	1	4

Fig 5: Prioritization on depending upon code coverage

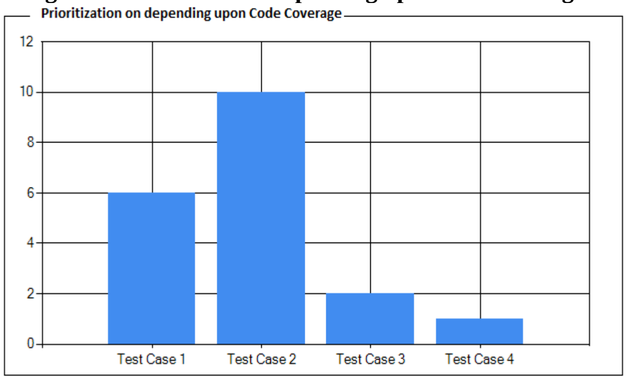
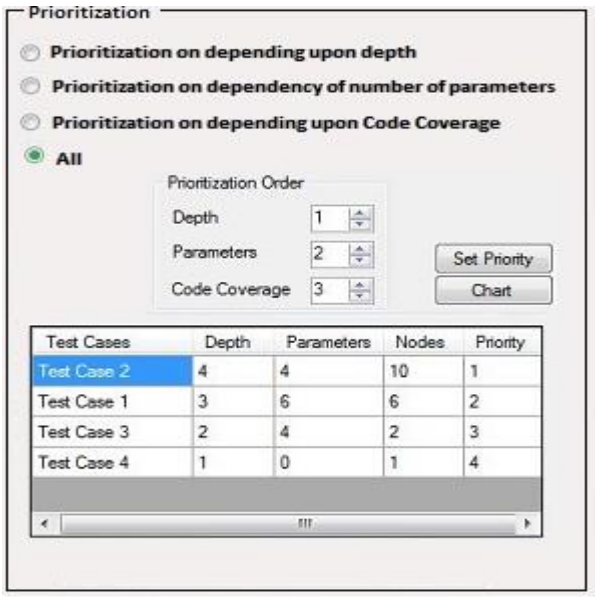


Fig 5.1 Results depending upon code coverage

4.4 Prioritization on Combination of all above Parameters

This prioritization technique will prioritize test cases on combining all the above three factors. In this case the priority can be changed depending upon the required conditions by changing the levels. This case sets priority by involving above three levels and is represented in Fig. 6 and Fig. 6.1.


Prioritization
Prioritization on depending upon depth
Prioritization on dependency of number of parameters
Prioritization on depending upon Code Coverage
Set Priority
Chart

● All

Prioritization Order

Depth	1
Parameters	2
Code Coverage	3

Test Cases	Depth	Parameters	Nodes	Priority
Test Case 2	4	4	10	1
Test Case 1	3	6	6	2
Test Case 3	2	4	2	3
Test Case 4	1	0	1	4

Fig 6: Prioritization on combination of parameters

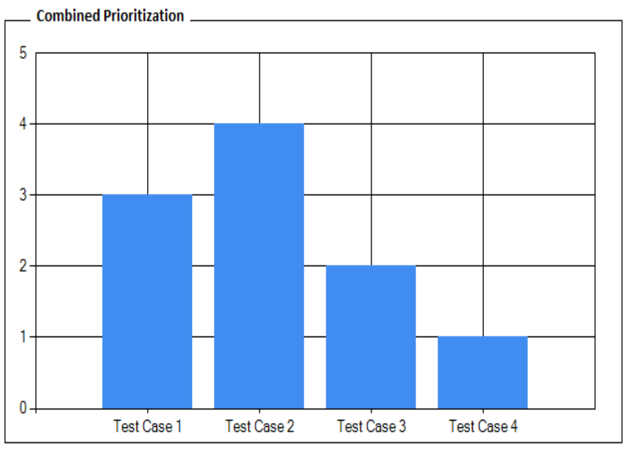


Fig 6.1 Results on combined factors (depth, parameter, code coverage)

5. CONCLUSIONS

This technique helps to improve the rate of fault detection and to find errors in early stage using test ordering method using test case prioritization.

This technique helps in finding critical areas that are important and should be tested first so as to early defect detection, reduce the cost of testing, saving time and making system more efficient. This technique first generates test cases and then assigns priority to each and every test case from the test case suite and then prioritizing these test cases in descending order of priority. Graphically results obtained helps to visualize user easily to find which test case is prioritized more so as to improved rate of detection of faults in comparison to random ordering of test cases.

This technique works as a generic tool for UML - Sequence diagram. Our proposed technique could be applied to other UML diagrams for generating and prioritizing test cases as a future work.

6. REFERENCES

[1] Praveen Ranjan shivastava and Tai-hoon Kim , "Application of Genetic Algorithm in Software Testing", International Journal of Software Engineering and its Application, vol.3, No.4, 2009 pp. 87-96.

[2] Sangeeta Sabharwal, Ritu Sibal and Chayanika Sharma, "Applying Genetic Algorithm for Prioritization of Test Case Scenarios Derived from ML Diagrams", International Journal of Computer Science, Vol. 8, May 2011, pp. 433- 444.

[3] R. Kavitha, Dr. N. Suresh Kumar, "Requirement Based Test Case Prioritization" IEEE, 2010.

[4] S. Elbaum, Malishevsky, G. Rothermel, "Prioritizing test cases for regression testing", ACM International Symposium on Software Testing and Analysis 25(5) 2000.

[5] Monalisa Sharma and Rajib Mall, "Automatic Test case Generation from UML Models", 10th International Conference on Information Technology, 2007, pp. 196-201.

[6] M. Sharma and D. Kundu, "Automatic Test case Generation for UML Sequence Diagram", IEEE 15th International Conference on Advance Computing and Communication, 2007, pp. 60-75.

[7] G. Rotherrmel and M. Harrold, "Selecting tests and identifying coverage requirements for modified software", ACM International Symposium on Software Testing and Analysis, Seattle, 1994.

[8] Praveen R Srivastava, "Test Case Prioritization", Journal of Theoretical and Applied Information Techonology, 2008.

[9] M. Khandai, Arup Acharya and Durga P Mohapatra, "A Novel Approach of Test Case Generation for Concurrent Systems Using UML Sequence Diagram", IEEE, 2011, pp. 157-161.

[10] Sujata, Mohit Kumar and Dr. Varun Kumar, "Requirement based Test Case Prioritization using Genetic Algorithm", International Journal of Computer Science and Technology Vol. 1,Issue 2, December 2010, pp. 189-191.

[11] S. Elbaum, A. Malishevsky, and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies", IEEE Transaction on Software Engineering, Vol.28, No.2, pp. 159-182, 2002.

[12] Aritra Bandyopadhyay and Sudipto Ghosh, "Using UML Sequence Diagrams and State Machine for Test Input Generation", 19th International Symposium on Software Reliability Engineering ISSRE, 2008, pp. 309-310.

[13] M. Prasanna and K.R. Chandran, (2011) "Automatic Test Case Generation for UML Object Diagram using Genetic Algorithm", Int. J. Advance. Soft Computer Application., Vol.1, No.1.