INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

AN INTEGRATED MODEL BASED TEST CASE PRIORITIZATION USING UML SEQUENCE AND ACTIVITY DIAGRAM

Lokesh Kumar Rathore[1], Neelabh Sao[2]

[1]M.Tech Scholar (Software Engineering), Rungta College of Engineering & Technology Bhilai,
lokesh.kr.rathore @gmail.com,
[2]Assistant Professor, Department of Computer Science & Engg, Rungta College of Engineering &Technology Bhilai,
neelabhsao@gmail.com,
Author Correspondence: Rungta College of Engineering & Technology Bhilai, 9893871459,
lokesh.kr.rathore@gmail.com

Abstract: - In light of quick addition in size and versatile nature of programming applications, more emphasis is given towards object oriented programming framework, which has any kind of effect to reduction programming cost and construct programming persistent quality, and convenience. In the meantime, presentation of object oriented arrangement and execution methodology draws out some new troubles for programming testing. A couple highlights of object oriented technique like polymorphism, inheritance, dynamic binding and so on makes certain troubles in programming testing methodology. To test such object oriented software system from their use code is to a great degree an incredible procedure due to the deferent characteristics of object oriented strategy. Model based testing of this programming strategy can be productive to recognize the exception in the design stage itself, so that these errors don't spread to distinctive periods of software development life cycle.

Software Testing and Maintenance is the crucial phase of software development life cycle (SDLC). More than half of the total development cost of a software system is associated with this phase. As the size and complexity of software get increases testing of software also becomes complex and takes more time and cost. Therefore it is required to detect faults as earlier as possible to maintain the software quality. Test cases or scenario prioritization is one of the effective techniques to find error earlier. In this paper, we proposed a combined approach to generate and prioritize test cases using UML Sequence and Activity diagrams. Before coding and other required activities of huge programming tasks have to perform the outlining of programming applications known as modelling. The objective of the project is effective use of UML model for test case prioritization with high fault detection rate and higher testing coverage criteria to reduce time and cost of software testing.

Keywords*:* Test Case Prioritization, Activity Diagram, Sequence Diagram, Software Testing.

1. Introduction

These days, because of fast increment in size and many-sided quality of programming applications, more accentuation is given towards object oriented software system, which makes a difference to decrease programming cost and build programming unwavering quality, and ease of use. At the same time, presentation of object oriented configuration and execution approach draws out some new difficulties for programming testing. Software Testing is the procedure of checking a framework or its constituents to perceive that it fulfils the predetermined necessities appropriately or not. To test software it is required to run it so as to discover missing necessities, occurrence of holes, and blunders if any in speak to the genuine basics. Before coding and other required activities of huge programming tasks have to perform the outlining of programming applications known as modelling. [1].

Modelling is the designing of software applications prior to coding and its necessary activity of large software projects. The UML (Unified Modelling Language) of OMG (Object Management Group) is visualization, specification, and documentation of software systems models, along with their design and structure. UML models are an important information source for test case formation and it is independent of methodology. UML can be used to express the results of design and analysis, regardless of the methodology used. The transformation of UML model from one tool into another tool or into a repository, or for refinement another OMG standard XMI (XML Metadata Interchange) can be used [4].

1.1 Model based test cases prioritization

In this test case prioritization technique model of a framework is utilized to schedule the test cases. Framework models are utilized to catch a few parts of the framework conduct. The prioritization of test cases with model may enhance the early error discovery when contrasted with the code based test case prioritization. Model based test prioritization may be a modest option to the current code based test prioritization systems. Be that as it may, model based test case prioritization may be touchy to the right/wrong data gave by the analyzers/developer. Hence forth demonstrate based prioritization of test case is the best one contrasted with code based prioritization.

1.2 Related UML diagrams

UML is a dialect for pointing out and not a system or method. The UML is utilized to characterize a software system; to detail the curios in the system, to report and develop it is the dialect that the blueprint is composed in. The UML is a widely used general purpose modelling language in software engineering, which is intended for defining, envisioning, developing, and archiving the software system artefacts. UML 2.0 characterizes diverse sorts of diagrams, divided into three categories:

- Structure Diagrams incorporate the Object Diagram, Class Diagram, Composite Structure Diagram, Component Diagram, Package Diagram, and Deployment Diagram.
- Behaviour Diagrams incorporate the Use Case Diagram (utilized by a few procedures while gathering requirements); Activity Diagram, and State Machine Diagram.
- Interaction diagram all got from the more general Behaviour Diagram, incorporate Communication Diagram, Sequence Diagram, Timing Diagram, and Interaction Overview Diagram.

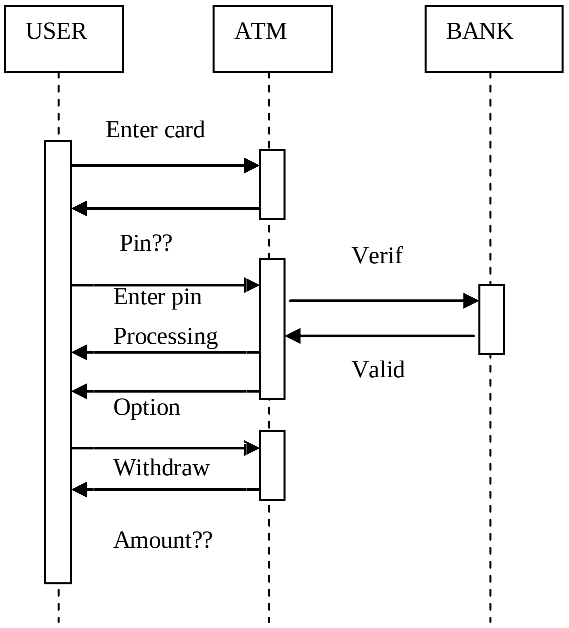

Figure 1: Sequence Diagram for ATM withdrawal

Sequence diagram or chart is the most widely recognized sort of interaction diagram, which concentrates on the message exchange between quantities of life savers. An UML sequence graph is an association outline that catches time subordinate (temporal) successions of collaborations down between items. They demonstrate the sequential grouping of the messages, their names and reactions and their possible contentions [4].

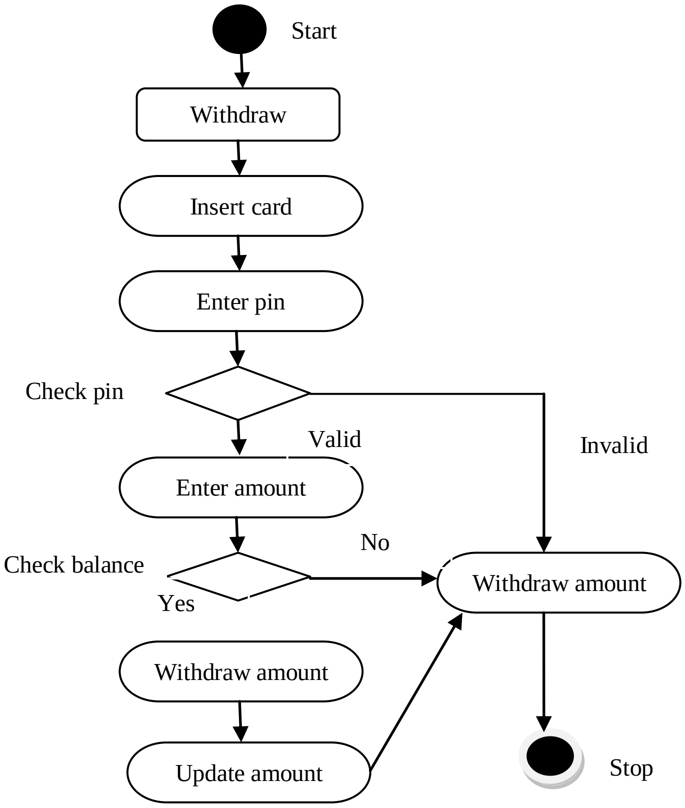

Figure 2: Activity Diagram for ATM withdrawal

Activity diagram is an alternate essential chart in UML to define dynamic parts of the system. Activity chart is essentially a flow diagram to show the stream structure one action to an alternate movement. Action graphs depict the work process conduct of the framework. These are like state diagram on the grounds that exercises are the condition of doing something. The simplest approach to imagine an activity chart is to think about a flowchart of a code. Activity diagrams accentuate the exercises of the object or a gathering of objects, so it is the ideal one to depict the realization of the operation in the design stage [4].

2. Related work

In [1] Swain R. K. et al (2013) has exhibited an incorporated methodology and a prioritization procedure to produce test scenarios from UML activity diagrams and communication at cluster level. In this methodology, first they build a tree representation of communication diagrams, and activity diagrams and convert them into an intermediate tree. At that point complete a post order traversal of the built tree for selecting restrictive predicates from the intermediate tree.

After this they use an algorithm to generate test scenarios and then prioritize those test scenarios. This strategy does not create redundant test scenarios and hence saves the time the time which can be utilized for testing the more basic components. This methodology is totally in light of the structural parts of the communication and activity diagrams. This method is in view of prioritization on the unpredictability of constructs in the activity diagrams and communication and it gives approx 60% average percentage fault detection rate.

In [3], Mohanty S., et al (2011) has proposed another prioritization system to organize the experiments to perform regression testing for Component Based Software System. The parts and the state changes for a segment based programming system are shown by UML state diagram which are then changed over into Component Interaction Graph (CIG) to define the interrelation among segments. The prioritization calculation takes this CIG as data along with the old test cases and produces an organized test suit considering aggregate number of state changes and aggregate number of database access.

In [4], Misra S. K and Mohapatra D. P, (2014) has presented a novel methodology for test scenario construction using UML sequence diagram focusing the pieces, nesting of fragments and control flow primitives display in sequence diagram. The strategy first creates a intermediate diagram called Sequence Control Flow Graph (SCFG) from the XMI representation of UML sequence graph. At that point they examine the control flow data, message flow and the fragment structure, it produces test scenarios, for different use case display in a system.

This approach is a completely methodical understanding of control flow data for different fragments. In this manner this methodology utilizes primitives of these control flow for test scenario creation in automated way. The test scenarios therefore created are suitable for useful testing an interaction and scenario faults.

In [5], Khandai M. and Acharya A. A., Mohapatra D. P., (2011) has introduced new way for test cases creation for parallel system with the assistance of UML Sequence Diagram. The Sequence Diagram is changed into a Concurrent Composite Graph (CCG) and then CCG is navigated by DFS and BFS (Breath-First-Technique) to produce test cases for concurrent systems.

This mechanism keeps away from test blast by taking care of the deadlock and synchronization issues of concurrency. The test cases are valuable for recognizing situation, operational and communication faults in case of concurrent systems.

The model based test case prioritization is developing system in test prioritization for testing the product in less time with upgraded shortcoming discovery rate.

3. Proposed method

Our proposed works provide the effective test case prioritization mechanism with minimum time and cost of testing the software application. We proposed a combined approach to generate and prioritize test cases using UML Sequence and Activity diagrams.

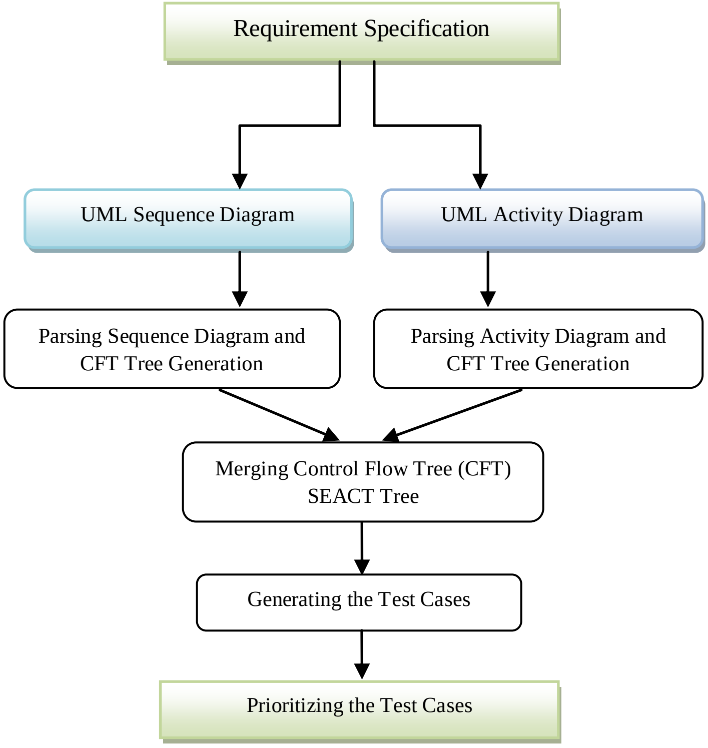
Requirement Specification
UML Sequence Diagram
UML Activity Diagram
Parsing Sequence Diagram and CFT Tree Generation
Parsing Activity Diagram and CFT Tree Generation
Merging Control Flow Tree (CFT) SEACT Tree
Generating the Test Cases
Prioritizing the Test Cases

Figure 3: proposed methodology

The objective of the project is effective use of UML model for test case prioritization with high fault detection rate and higher testing coverage criteria to reduce time and cost of software testing. Our proposed approach is shown in fig 6. The detailed view of our proposed methodology is illustrated along with the case study of travel agency a reservation application in the following phases:

3.1 Constructing Sequence diagram and Activity diagram of system under test

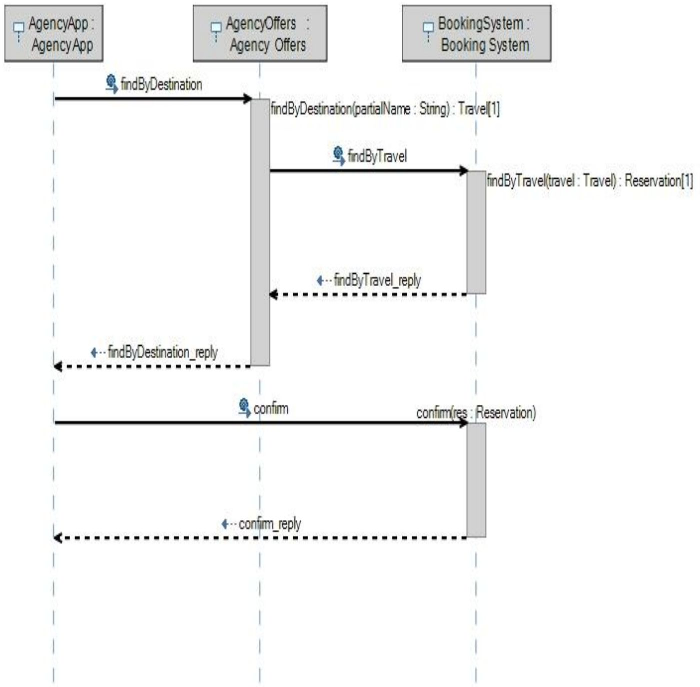
AgencyApp : AgencyApp
AgencyOffers : Agency Offers
BookingSystem : Booking System
findByDestination
findByDestination(partialName : String) : Travel[1]
findByTravel
findByTravel(travel : Travel) : Reservation[1]
findByTravel_reply
findByDestination_reply
confirm
confirm(res : Reservation)
confirm_reply

Figure 4: Sequence diagram for Travel reservation agency

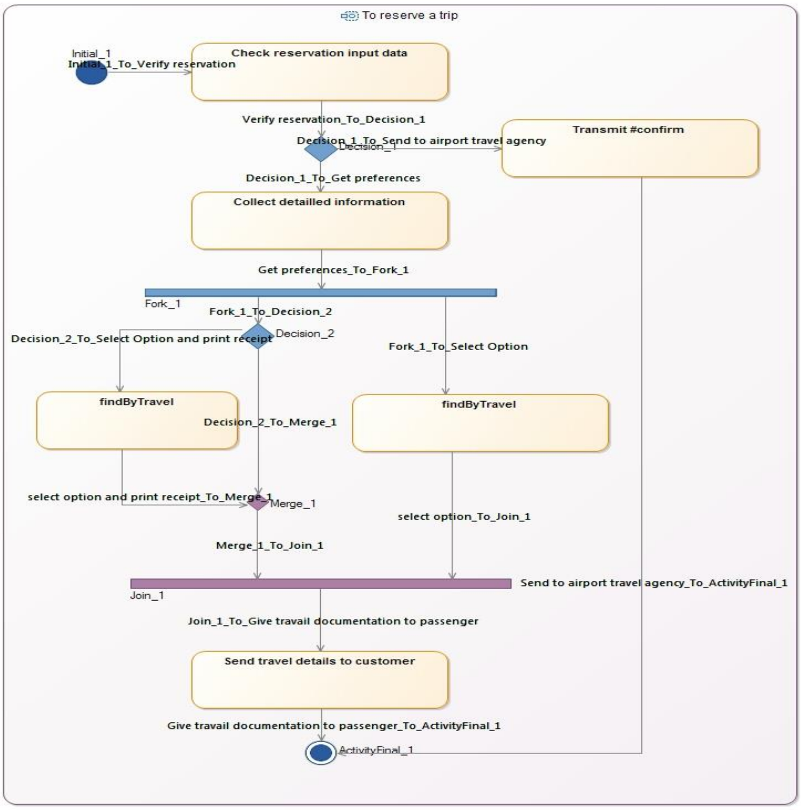
To reserve a trip
Initial_1
Initial_1_To_Verify reservation
Check reservation input data
Verify reservation_To_Decision_1
Decision_1_To_Send to airport travel agency
Decision_1
Transmit #confirm
Decision_1_To_Get preferences
Collect detailled information
Get preferences_To_Fork_1
Fork_1
Fork_1_To_Decision_2
Decision_2
Decision_2_To_Select Option and print receipt
Fork_1_To_Select Option
findByTravel
Decision_2_To_Merge_1
findByTravel
select option and print receipt_To_Merge_1
Merge_1
select option_To_Join_1
Merge_1_To_Join_1
Join_1
Send to airport travel agency_To_ActivityFinal_1
Join_1_To_Give travail documentation to passenger
Send travel details to customer
Give travail documentation to passenger_To_ActivityFinal_1
ActivityFinal_1

Figure 5: Activity diagram for Travel reservation agency

A very first step in our proposed method is modelling the system under test. We create the sequence diagram and activity diagram shown in Fig 7 and Fig 8, as our UML model for the given system for which we use UML designer; a modelling tool .

3.2 Analyzing UML diagrams and Designing control flow tree CFT

The second step of our project is to parse both the sequence diagram and activity diagram and extract the control flow information from them as shown in block diagram in Fig 6.

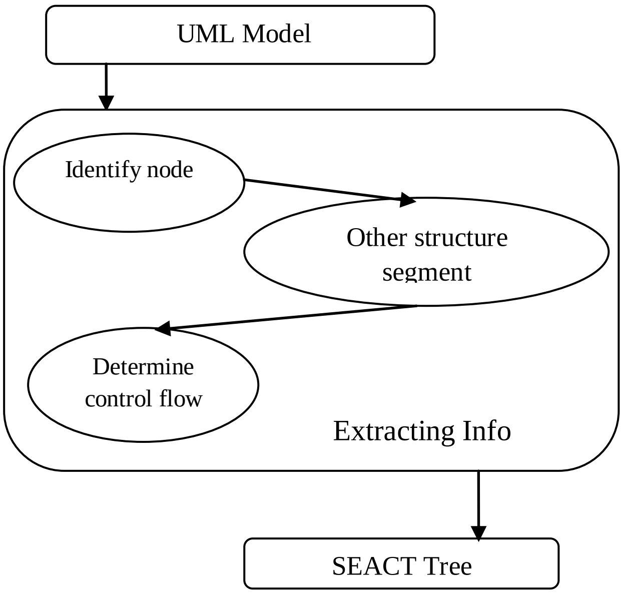

Figure 6: Extracting control flow scenario from UML Model

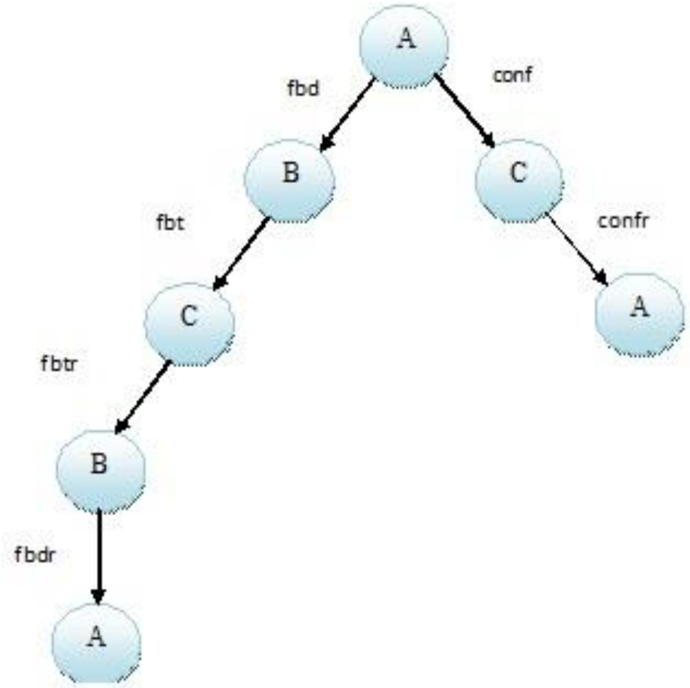

Figure 7: SeqCFT Tree for Travel reservation agency

After extracting information from UML model we transform sequence diagram and activity diagram into CFT and named them SeqCFT and ActCFT respectively which are shown in Fig 7 and Fig 8 correspondingly.

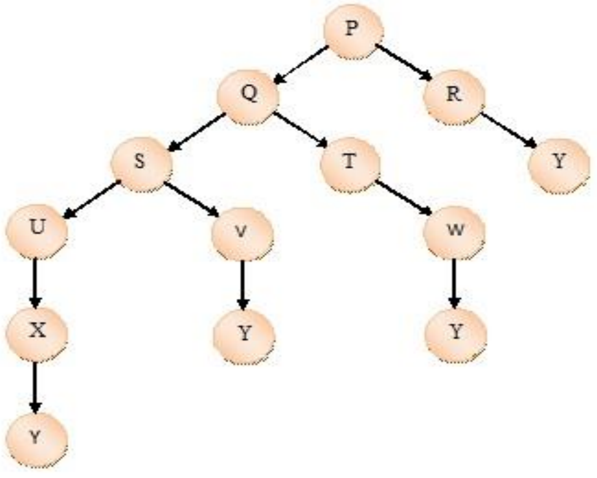

Figure 8: ActCFT Tree for Travel reservation agency

3.3 Merging CFTs and creating SEACT Tree

We have combined message flow sequence in sequence diagram with the control activities in the activity diagram for test scenario creation. To do this, we joined SeqCFT of Sequence diagram with the ActCFT of activity diagram into a solitary CFT namely SEACT. Fig.9 illustrates SEACT tree created from sequence and activity diagrams described in Figs. 7 and 8, respectively.

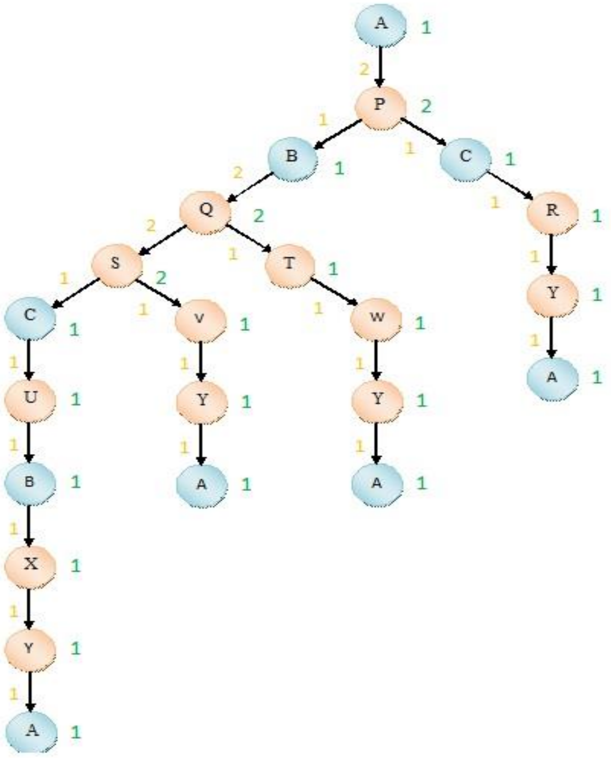

Figure 9: SEACT Tree for Travel reservation agency

3.4 Generating test cases

The next step is to determine different control flow sequences which will be our desired test scenarios/cases. To identify test cases here we use modified DFS i.e. Depth First Search Algorithm on SEACT Tree as given below:

Algorithm for Test Cases generation

```
Input: SEACT tree, in which every node S includes the following info:{i/p, o/p, weight}
Output: Set of test cases (STC)
1. i := 1
2. S := startnode //S = current node
3. repeat
4. Create Pi = φ //  Pi = i^th test case
5. while (S != L) do //where L = leaf node
6. label S as discovered
7. Pi = Pi ∪ S
8. S = C //C = undiscovered child node of S
9. end while
10. i = i + 1
11. S = unlabelled child of the last branching node
11. until all node are labelled
13. N := i //N = total number of test cases
14. STC := $\bigcup_{i}^{N} Pi$ //STC = set of test cases
15. exit
```

3.5 Prioritizing the test cases Parsing UML Diagrams and generating SEACT Tree

The test cases generated in the previous step are prioritized in this step as per the flow of control based on behavioural aspects of the software application which is recognized SEACT of UML model. The Following mathematical calculations are performed:

- Assign weights to the nodes of the SEACT Tree

Weight (n)= no. of children of n; n: node

- Assign weights to the edges of the SEAC Tree

Weight (e) = $(n_i)_{in}$ x $(n_j)_{out}$; e: edge

Where $(n_i)_{in}$ is the number of incoming edges of node n_i and $(n_j)_{out}$ is the number of outgoing edges of node nj and e is the edge connecting n_i and n_j.

- Prioritize Scenarios/Cases

Considering the weights of each path p, we prioritize the scenarios for the corresponding paths in order of decreasing weights of p which is calculated as follows in eq. (1).

$$Weight(p) = \sum_{i=1}^{n} Weight(n_i) + \sum_{j=1}^{m} Weight(e_j)$$

(1)

4. Result and discussion

The proposed work is give the persuasive test prioritization instrument with least time and expense of testing the product application as our proposed strategy deals with both temporal communication as well as relational behaviour of the system model. The result of execution of prioritized test cases for given input UML model of Travel Reservation System Case study consisting of prioritized test cases and for evaluation of our approach APFD (Average Fault Detection) Rate is used to show how many test cases run successfully, or failed or ignored which is given in eq. (2).

$$APFD = 1 - \frac{TF1 + TF2 + .. + Tfm}{n * m} + \frac{1}{2 * n} \qquad (2)$$

Where T is set of test cases, m: number of faults enclosed in system under test, n is the total number of test scenarios and TFi: the position of the first test in T that depicted fault i.

In our case study of travel reservation agency application we have assume 5 faults positioned at SEACT node Q, S, W, R and X and named as F1, F2, F3,F4, and F5 respectively. We have calculated APFD value for this in both non-prioritized and prioritized case as follows:

- For Non-prioritized test cases:

  APFD=1 - ((2+3+2+1+4) / (4*5)) + 1/ (2*4) = 0.525

- For Prioritized test cases:

  APFD=1 - ((1+1+2+4+1) / (4*5)) + 1/ (2*4) = 0.675

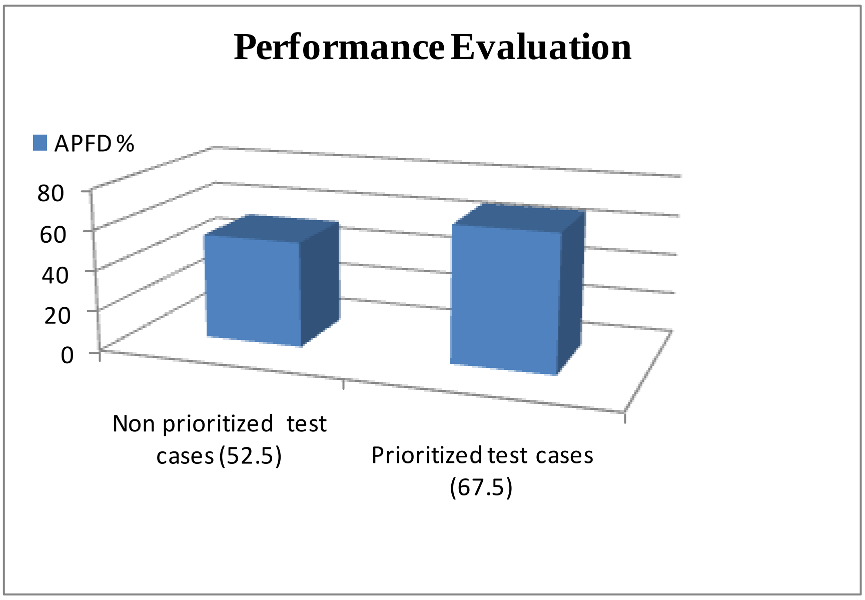

Figure 10: Performance evaluation of our SEACT approach

From the above evaluation it is clear that APFD value attains in our approach is for prioritized test cases is greater than that of non-prioritized test cases. Thus our approach makes an effective test case prioritization model as per our case study.

4. Conclusion

The model based test case prioritization is the emerging technique in test case prioritization for testing the software in less time with optimized fault detection rate. In our determination of leaving testing strategies exploration is centred on strategies in light of UML models. For deciding the adequacy of prioritization strategies measurement (APFD) is utilized. There is great scope regarding research in understanding the ideas of model based strategies and conduct of segments, associations and similarity of parts.

Our future work would be, to take into consideration to make completely automated tool for our approach of model based testing. We will also try to integrate code level testing in our method of testing as a sandwich approach for enhancing performance. There are also scope to take other combination of UML model to get better result for optimized usage of resources such as cost and time of testing.

REFERENCES

[1] Acharya A. A, Mohapatra D. P., Panda N., (2010), "Model Based Test Case Prioritization for Testing Component Dependency in CBSD Using UML Sequence Diagram" (IJACSA) International Journal of Advanced Computer Science and Applications, December 2010, Vol. 1, No. 6.

[2] Beilei Liang, Pan Liu, Huaikou Miao, (2013), "Scenario Specification based Testing Model Generation," IEEE 12th International Conference on Computer and Information Science (ICIS), IEEE/ACIS, 2013, pp 335-340.

[3] Catal C., Mishra D., (2012), "Test case prioritization: a systematic mapping study" Springer Science+Business Media, LLC 26, july 2012.

[4] Faria J. P., Paiva A. C. R., Zhuanli Yang (2012), "Test Generation from UML Sequence Diagrams," IEEE Eighth International Conference on the Quality of Information and Communications Technology, 2012, pp 245-250.

[5] Garg D., Datta A., French T., (2012), "A Two-Level Prioritization Approach for Regression Testing of Web Applications", IEEE 2012, 19th Asia-Pacific Software Engineering Conference, (Volume: 2), pp150-153.

[6] Garg D., Datta A., (2012), "Test Case Prioritization due to Database Changes in Web Applications", IEEE 2012, Fifth International Conference on Software Testing, Verification and Validation, pp 726-730.

[7] Gantait A. (2011), "Test case Generation and Prioritization from UML Models", IEEE Second International Conference on Emerging Applications of Information Technology, 2011, pp 345-350.

[8] Khandai M., Acharya A. A., Mohapatra D. P., (2011), "A Novel Approach of Test Case Generation for Concurrent Systems Using UML Sequence Diagram", 3rd International Conference on Electronics Computer Technology (ICECT), 2011, (Volume: 1), pp 157-161.

[9] Kundu D.. Samanta D., Mall R., (2012), "An approach to convert XMI representation of UML 2. x interaction diagram into control flow graph". ISRN Software Engineering 2012, pp. 1-22.

[10] Misra S. K, Mohapatra D. P, (2014), "A Novel Methodology For Test Scenario Generation Based On Control Flow Analysis Of Uml 2.X Sequence Diagrams", IJRET: International Journal of Research in Engineering and Technology, Volume: 03 Special Issue: 15, Dec-2014.

[11] Mahali P., Acharya A. A., (2013), "Model Based Test Case Prioritization Using Uml Activity Diagram And Evolutionary Algorithm" International Journal of Computer Science and Informatics, 2013 Volume-3, Issue-2, ISSN 2231 –5292.

[12] Mohanty H., Sapna P.G., (2009), "Prioritization of Scenarios based on UML Activity Diagrams," IEEE First International Conference on Computational Intelligence, Communication Systems and Networks, 2009, pp 271-276.

[13] Mohanty S., Acharya A. A., Mohapatra D. P., (2011) "A Model Based Prioritization Technique For Component Based Software Retesting Using Uml State Chart Diagram", in IEEE Electronics Computer Technology (ICECT), 2011 3rd International Conference (Volume: 2), pp 363-368.

[14] OMG (2005) Unified Modeling Language Specification, Version 2.0. Object Management Group. http://www.omg.org

[15] Panigrahi C. R., Mall R., "Model-Based Regression Test Case Prioritization", ACM SIGSOFT Software Engineering, November 2010, Volume 35, No 6.

[16] Sabharwal S., Sibal R., Sharma C., (2010), "Prioritization Of Test Case Scenarios Derived From Activity Diagram Using Genetic Algorithm", IEEE International Conference on Computer and Communication Technology (ICCCT), 2010, pp 481-485.

[17] Shanmuga Priya S., Sheba P. D., (2013), "Test Case Generation From Uml Models – A Survey", International Conference on Information Systems and Computing (ICISC-2013) Volume 3, Special Issue 1.

[18] Swain S. K., Mohapatra D. P,and Mall R.,,(2010), "Test Case Generation Based on Use case and Sequence Diagram", July 2010, Int.J. of Software Engineering, IJSE Vol.3 No.2.

[19] Swain R.K., Panthi V., Mohapatra D. P., Behera P. K. (2013), "Prioritizing test scenarios from UML communication and activity diagrams", Springer-Verlag London, 2013 Innovations in Systems and Software Engineering Volume 10, Issue 3, pp 165-180.

A Brief Author Biography

***1st Lokesh Kumar Rathore* –** M.Tech Scholar (Software Engineering), Rungta College of Engineering & Technology Bhilai, lokesh.kr.rathore @gmail.com

***2nd Neelabh Sao* –** Assistant Professor, Department of Computer Science & Engg, Rungta College of Engineering &Technology Bhilai, neelabhsao@gmail.com,