Minimization of Test Suites for Fuzzy Object-Oriented Database

Shweta Dwivedi
Assistant Professor
Dept. of Computer Science & Engineering,
Maharishi University of Information Technology,
Lucknow, INDIA

Santosh Kumar
Associate Prof. & Head
Dept. of Computer Science & Engineering
Maharishi University of Information Technology,
Lucknow, INDIA

ABSTRACT
Test suite optimization is an effective method that is used to minimize or reduce the time and cost of testing. There are several researchers and software professionals have used this method or techniques to enhance the correctness and effectiveness of test suites/cases. These approaches optimize test suite for a single objective but fuzzy logic with certain algorithm is used to optimize the test cases/suites for multi objective selection processes. Therefore, in the present work an approach of optimization or minimization of test suites is proposed for the designed fuzzy object-oriented database and also applied this approach to reduce the complexity of the designed database. The main objective of our approach is to minimize and find the test suite that is best optimal for multi-objective testing.

Keywords
FOOD, Test suites, Test suites optimizations, C-Means Clustering.

1. INTRODUCTION
Software testing is one of the important tasks for any software testers to test that the designed system is working properly and accurately. The designing of good test cases is very difficult and complex art. According to Patton R. [83] test cases are the specific inputs that you will try and the procedures that you will follow when you test the software. Therefore, a good test case is that helps to discover correct information along with the risk which is associated with the information. The test cases are designed and executed to find the bugs and make them fixed before deployment the software. There are different styles of testing is available (like Stress Testing Regression Testing Risk-based testing Domain testing etc.) to validate the developed software. In the current scenario, the regression testing is used to reduce the efforts of re-testing the software to find an appropriate and adequate testing coverage, although the regression testing is the process to re-test the designed system for ensuring that the previously validated system has no more errors. Any changes made in the software evolved huge number of test cases that resultant a time consuming, highly expensive and exhaustive process to re-test the system. However, the test cases are the collection or set of logic or condition through which the designed system is tested and determined that whether the system satisfies or meet the requirements and working correctly. Therefore, the minimization of test cases is one of the major challenges in the software development system. The test case reduction is one of the techniques that try to remove the redundant and unnecessary test cases.

As the testing process is on progress some new test cases may be added in regression testing, the resultant of this process the size of test suites is increased due to the integration techniques. Therefore, the increment of test cases makes the possibility to produce some redundant test cases also. Due to the limitation of the time and resources, the minimization or optimization techniques should be required to used and minimize the test suites/test cases.

2. RELATED WORK
Bhasin et al. [1] have proposed a work dwells on the power of fuzzy expert system to make decisions which are better than the normal expert systems. Mudgal [2] has proposed a new model for the minimization of test suite, which is based on the Boolean function simplification. Alakeel [3] has presented a novel approach for test cases prioritization using fuzzy logic for the purpose of regression testing programs with assertions. Haider et al. [4] have proposed an expert system that finds a tradeoff among the quality aspects, technique used and level of testing based on objective function defined by the tester, quite similar to human judgment using fuzzy logic based classification. Haider et al. [5] have concluded that ANFIS can be used to automate the optimization process. Shamshiri et al. [6] have reported an empirical study on the effects of using a genetic algorithm (GA) to generate test suites over generating test suites incrementally with random search, by applying the Evo Suite unit test suite generator to 1,000 classes randomly selected from the SF110 corpus of open source projects. Qiu et al. [7] have presented a qualitative analysis of the findings, including stakeholders, challenges, standards, techniques, and validations employed in these primary studies. Kumar and Bhatia [8] have proposed a methodology based on fuzzy clustering by which we can significantly reduce the test suite. Usaola et al. [9] have proposed an algorithm for reducing the size of test suites, using the mutation score as the criterion for selecting the test cases while preserving the quality of the suite. Singh and Shree [10] have added new test cases to the existing test suite to check the new functionality implemented in the release. Asoudeh and Labiche [11] have proposed a test suite generation technique from extended finite state machines based on a genetic algorithm that fulfills multiple (conflicting) objectives. Ma et al. [12] have investigated the use of an evolutionary approach, called genetic algorithms, for test-suite reduction. Selva kumar et al. [13] haveexamined the effectiveness of a test suite reduction process based on a combination of both concept analysis and Genetic algorithm. Dinca [14] has introduced the multi-objective test suite optimization problem for Event-B testing. Raamesh and Uma [15] have utilized a hybrid, multi-objective algorithm that combines the efficient

approximation of the evolutionary approach with the capability data mining algorithm to produce higher-quality test cases. Pattorn [16] has introduced the software testing techniques to test the designed software. Mondal and Tahbildar [17] have presented an approach showing that the novel optimal page replacement algorithm reducing the redundant test cases during retesting of modified object oriented program. Subhashini and Jeya Mala [18] have reduced the time spent in testing by reducing the number of test cases.

3. EXPERIMENTAL STUDY & RESULTS

An approach of test cases minimization for heart disease diagnosis is proposed, for that purpose a case study of hospital-based patient diagnostic system is taken and a diagnostic process is represented through the state chart diagram also generate some test cases for demonstrating that how the test cases are minimized.

3.1 State Chart Diagram

State chart diagram known as a state machine diagram in UML is used to design the different states of the system through which the classes passes from state to state where every class is assigned with a state. The state machine diagram is designed by the basic notations like filling the circle for the initial state, a hollow circle filled with the solid circle for final state, rounded rectangle showing the state and an arrow is used for transition of states. The figure 6.3.1.1 shows the state chart diagram of patient diagnostic process who suffered from heart attack problem. Therefore, it is assumed that Patient is the initial as well as final state who suffered from heart disease this state is equivalent to 'q_0' and the Patient goes to the Registration_ Desk for register himself where the patient get their number and goes to the concerned department where the Doctor diagnose the patient in details and send the patient to the concerned Ward to admit the patient if the condition is severe; these states of the system are equivalent to 'q_1', 'q_2' and 'q_3' respectively. If the condition of the patient is moderate then the patient take the prescription from the doctor and go home as well as the patient is declared fit by the Doctor and discharged from the ward i.e. the system reach to its final state this state of the system is 'q_0'.

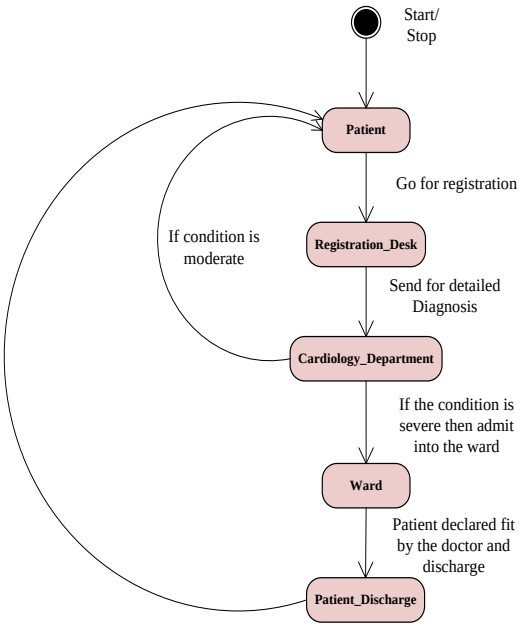

Fig 3.1: State Chart Diagram for Heart Patients Diagnosis

There are some tests cases are designed for the verification or correctness of the above designed state chart diagram which are described below:

Test Case 1

The Registration_Desk/MSW checks the condition of the patient. The Patient is not admitted into the ward if the condition of the patient is moderate.

Test Case 2

The Patient is found in severe condition by the doctor and referred the patient to admit into the ward.

Test Case 3

The Patient declared fit by the doctor and the patient discharged from the ward with medicines prescription.

3.2 Fuzzy Object-Oriented Database

The Fuzzy Object-Oriented Database is an extension of structured based database like relational database for handling the both crisp and vague information. There is a great need to represent the vagueness or impreciseness in today's several applications that are dealing with the different nature of the data. Basically, the fuzzy object-oriented database allows both the conceptual schema and the navigation into the database through a graphical interaction. Therefore, the fuzzy object-oriented database is designed to deal or manage incomplete information at the level of data representation and in queries and has been formalized within the framework of fuzzy set theory. In respect of fuzzy object-oriented database, it is possible that either it deals with various kinds of data richer than the traditionally managed by the pure object-oriented database. Therefore, fuzzy object-oriented database is a combination of both structured as well as object-oriented database where the users can manipulate and retrieve information by performing several queries as performed in the traditional query languages.

Unclear and inconsistent information is handled by the most promising database i.e. the fuzzy database. An extension of the fuzzy database is fuzzy object-oriented database (FOOD) that also deals with the vague or imprecise information as well as it supported the object-oriented programming concepts for storing and interrogating the vague information and turned

this vague information into crisp one. Therefore, a fuzzy object-oriented database is designed for the patient diagnostic system (PDS) of **"Heart Disease"** is represented in the table 3.2.2 and their Membership Function along with range value is represented into the table 3.2.1. Some fuzzy queries are performed, for that the fuzzy query approach is based on the fuzzy logic. Here a fuzzy object-oriented database is designed for the patient who suffered from the heart failure problem.

For fuzzy database a member function plays an important role; therefore, some major member functions are taken in context of patient heart failure like patient Age with its linguistic variable (Young, Mid, Old, Very Old), Cholesterol with linguistic variable (Low, Medium, High, Very High) and Blood Pressure with linguistic variable (Low, Medium, High, Very High) along with their fuzzy set range values are represented in table 3.2.1.

Table 3.2.1. Membership Function for Heart Disease and Linguistic Terms along with Fuzzy Range Values

Input Field	Linguistic Terms	Range Value	Input Field	Linguistic Terms	Range Value	Input Field	Linguistic Terms	Range Value
Age	Young	<35	Cholesterol	Low	<195	Blood Pressure	Low	<127
	Mid	35-45		Medium	195-250		Medium	127-160
	Old	45-58		High	250-310		High	160-185
	Very Old	58>		Very High	290>		Very High	180>

Table 3.2.2. Fuzzy Object-Oriented Database for Heart Disease Diagnosis

Results | Messages

	P_ID	Name	Doctor_ID	Gender	Age	Weight	Disease	Range_Blood_Pressure	Range_Colestrol
1	1001	MR. ANURAG SHARMA	D0001	Male	45	76	Chest Pain	142-150	217-307
2	1002	MR. JAGDISH	D0002	Male	67	67	High Blood pressure	155-197	>300
3	1003	MRS. CHANDAWATI	D0003	Female	78	57	Chest Pain	155-197	>300
4	1004	MASTER JAINUL	D0004	Male	23	60	Chest Pain	127-155	197
5	1005	MR. SANTOSH KUMAR	D0005	Male	35	56	Chest Pain	127-155	200
6	1006	BABY RADHA	D0006	Male	12	15	chest pain	127-140	<190
7	1007	MR. JITIN WADHWANI	D0007	Male	69	55	Chest Pain	155-200	>300
8	1008	MRS. KIRAN	D0008	Female	25	45	Chest Pain	127-155	190
9	1009	MRS. USHA MISHRA	D0009	Female	56	78	Chest Pain	155-200	>200
10	1010	MISS. SUHASI	D0010	Female	42	55	High Chest Pain	155-200	>300
11	1011	MRS. NIRMALA	D0011	Female	55	46	Chest Pain	155-197	>200
12	1012	MISS. SHAHEEN BANO	D0012	Female	65	45	High Blood Pressure	155-200	>300
13	1013	MISS. ROHINI	D0013	Female	76	50	High Chest Pain	155-200	>300
14	1014	MR. RAMESH	D0014	Male	51	66	Chest Pain	155-197	>200
15	1015	MASTER ANKIT	D0015	Male	66	56	High Blood Pressure	155-200	>300
16	1016	MR. RAM SWAROOP	D0016	Male	70	63	High Chest Pain	155-200	>300
17	1017	MRS. PYARA	D0017	Female	71	36	High Blood Pressure	155-200	>300
18	1018	MR. TULAI GAUTAM	D0018	Male	78	38	Chest Pain	155-197	>200
19	1019	MRS. SHAYRA	D19	Female	62	20	High Chest Pain	155-200	>300
20	1020	MR. SHIV DULARE	D0020	Male	49	65	High Blood Pressure	155-200	>300
21	1021	Mrs. Shamila singh	D0022	Female	63	56	Chest Pain	155-197	>200

Query executed successfully. (loc

3.3 Test Suites for Fuzzy Object-Oriented Database

The test suites are a collection of several test cases and the test cases are the set of conditions that are used to test the software or any developed software system by the software professional like tester who will test the designed system whether it satisfying the requirement or work properly for which the system is designed. Through the test case writing software professional can also find the problems occur in the developing system. There are several test case tools available through which a software professional can write test cases. A sample test case format is represented in the table 3.3.

Table 3.3. Sample Test Case Format

Test Suite ID	Test suite to which these test cases belongs
Test Case ID	ID of the test case
Objective/Summary of Test Case	
Related Requirement	Test case requirement ID
Prerequisites	Preconditions that must be fulfilled before to executing the test
Test Procedure	Procedure for test case execution
Test Data	Required test data that is used during the test case executed
Expected result	Expected outcome
Actual Result	Actual outcome
Status	Pass the test/fail the test
Remark	Any remark by the tester

Created/run by	Person who write test cases
Date of Creation	Test case creation date
Executed by	Executed person
Date of Execution	Test case execution date
Test Environment	Environment (Hardware/Software/Network) at which tests were executed

Similarly, the software professionals and the software testers made several test suites that include several test cases and verified that the designed system is satisfied the all requirement specification and working properly.

3.4 Minimization/Optimization Techniques for Test Suites

The collection of several test cases which are performed by the software professionals or researchers to test a software system is called a test suite. Test case writing for any software system or application is a skill which can be achieved through the experience of several years and study. An effective test cases is written by the experienced software testers. As the well experienced software testers or professionals write several test cases or bunch of test cases for a software application, there certain chances for redundant test cases which affect the cost, time and size of the test suite. Therefore, software professionals need to reduce these test cases to save the time and cost. As the name said the test suite reduction is a technique to reduce the similar or redundant test cases and also reduce that test cases which are not relevant or have less important for the software system. Therefore, the minimization of test suite for the system or software application can be formally stated as follows.

Suppose:

(1) A test suite T consists of several test cases {t1, t2, t3, . . .,t_n}.

(2) Testing requirements are the one that must be satisfied the desired testing coverage of any software system or applications. Here a testing requirement set {r1, r2, r3, . . ., rn}is taken that must be satisfied to provide the desired testing coverage of the program.

(3) The subset of test suite set is {T1, T2, T3, . . ., Tn} of T, in which each T associated with each of the r_n such that each test case t_n belongs to T_n that satisfied r_n.

Though the software professionals design and write the test cases to verify the designed system's correctness but they write bunch of test cases that includes lots of tests which is very exhaustive, time taking and cost effecting when the software tester executes these tests. Therefore, software professionals need to optimize or minimize these test case to reduce the time and coast of testing. An optimization technique is one through which minimum test cases are used to produce the correct and an effective solution of the problem.

Here a clustering technique is used for minimized the test suites for that purpose test cases are written or created through any one of the testing methods like white-box testing/black-box testing measured the test data and then reduce the test cases by using the clustering technique. Architecture for test suite minimization is represented in figure 3.4 where it is shown that the system transformed into the graphical form which is represented into the figure 3.1 where the path between to states is covered by the test data.

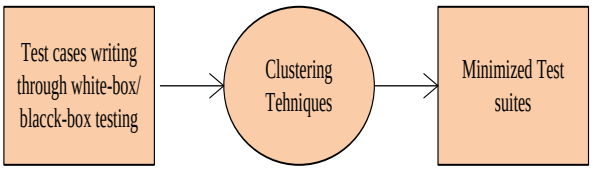

Figure 3.4: Architecture or test suite minimization

Test suite minimization architecture demonstrate that the test case data is measured by any one of the testing techniques of the system which is being tested and converted into the control flow graph (in this chapter graphical form of the system represented through the state chart diagram in figure 3.1) where the path between states covered by the test data. Lets us consider an algorithm for heart disease diagnosis and data set for measuring the heart disease is represented into the table 3.4.

Table 3.4.1: Data set for Measuring the Heart Disease

Key_Value	Test_Value	Input_Value
P_ID	Test value range for heart disease (no heart disease if diameter narrowing i.e. 0%-50% and heart disease confirmed if diameter narrowing range is >50%)	**Age** **Gender:** (value 1: Male; value 0: Female) **Chest Pain Type:** (value 1: Normal, value 2: Moderate 3: Severe) **Blood Pressure:** (mm Hg on admission to the hospital) **Cholesterol** (mg/dl)

ALGORITHM:

Step1: Input Value: Gender, Age, Chest_Pain_Type, Blood_Pressure, and Cholesterol.

Step2: Each input variables is fuzzy and associated with the membership function.

Step 3: Calculate each membership function for fuzzy variable.

If Gender = Male/Female, Age = 45-75, Chest_Pain_Type = 2/3(Moderate/Severe), Blood_Pressure (155-197) and Cholesterol (217-307) then the heart disease condition is confirmed.

Step 4: Repeat the step 3 for several time for different input values to confirm the heart failure condition.

Step 5: Exit

3.5 Control Flow Diagram

A diagram that describe the flow of controls on any process or system that consists of subdivision to represent the process to compute the result in a sequential form of steps. Here geometrical figures like Square, Diamond, oval etc. to represent operations of data and arrows are used for the path that shows the flow of controls. There are various types of control flow diagrams such as process control flow diagram, quality control flow diagram, Configuration control flow diagram etc. therefore; a control flow diagram for measuring

the heart disease is represented in the figure 3.5 which is a directed graph consisting of vertices and edges where every vertex is the state and the directed arrow represent the flow of control while P1, P2, P3 and P4 represent the path of the control flow.

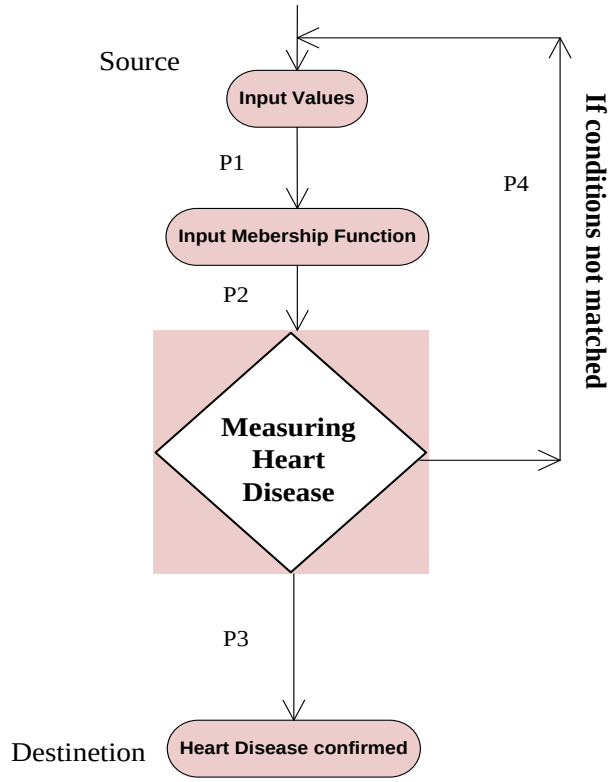

Fig 3.5: Directed Graph for Heart Disease Diagnosis

Table 3.5. Test Case and Path for Heart Disease Diagnosis

Test Case ID	Input Value	Range Value	Expected Result	Path
t_1	Chest Pain, BP	Moderate, 155-199	Heart disease Confirmed	Source-P1-P2-P3-Destination
t_2	Age, Chest Pain, Cholesterol	75, severe, 199-297	Heart disease confirmed (heart failure)	Source-P1-P2-P3-Destination
t_3	Age, Chest Pain, BP	25, moderate, 127-140	No heart disease	P1-P2-P4
t_4	Chest Pain	Severe	No heart disease	P1-P2-P4

The test cases and their paths for heart disease diagnoses are represented in the above table 3.5 which are clustered and minimize through the c-means clustering method. The data sets/objects are grouped into the similar cluster through the clustering analysis and different data objects puts into dissimilar group. Therefore, the c-means clustering is explained here for the heart disease diagnosis.

3.6 C-Means Clustering

Clustering is the process of collect and arranges items into groups on the basis of their similarity. There are different types of clustering algorithms such as K-Means Clustering and C-Means Clustering; here the c-means clustering algorithm is used for minimize the test suites. Therefore, the fuzzy c-means clustering algorithm contains the groups of items that may belongs to more than one groups and the degree of membership function for each item is given by a probability distribution over the clusters. The c-means clustering algorithm is very useful where the numbers of clusters are predetermined. Therefore, the algorithm put the data to any one of the clusters.

Due to the non-decidability of absolute membership function of the data which belongs to a given cluster it is different from other clustering algorithm. It measured the degree of membership function. The fuzzy c-means clustering is very fast and accurate in calculating the absolute membership function because there are several numbers of iterations required to accomplish a particular clustering exercise corresponds to required accuracy. The equation (1) shows an objective function which is minimized in each iterations of the c-means clustering algorithm:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{C} \delta_{ij} \parallel x_i - c_j \parallel^2 \quad (1)$$

Where:

N → number of data points
C →number of clusters required
c_j→Centre vector for cluster j,
δ_{ij}→Degree of membership for the i^{th} data point x_i in cluster j
$\parallel x_i - c_j \parallel$→Measures the similarity/closeness of the data point x_i to the Centre vector $c_j$of cluster j.

In each iteration, the algorithm maintains a center vector for each of the clusters. Therefore, the main objective is to define c clusters for each data points and placed these clusters in smarter way that produces a different result. in the proceeding step take each data point belonging to the data set that is associated with the nearest cluster. The first step of clustering is completed when there is no data point is remaining and at this stage need to re-calculate the c new clusters as barycentre of the result of the previous step. As the c new cluster found, a new binding has been developed between the data point and the nearest cluster. For further proceedings a loop is generated and noticed that these cluster moving their locations step by step until no more changes can be done. Therefore, the objective of this algorithm is to minimize the objective function shown in the equation (1).

4. CONCLUTIONS

Many researchers have proposed test suite optimization or minimization techniques to reduce the redundancy and duplicity in test suites that increase the cost and time of testing. The test suites minimization technique is also studied

by the concept of regression testing for the designed fuzzy object-oriented database and the test suites are minimized against some criteria that proved the resulting test suite is capable to find almost entire range of bugs. Minimization or optimization of test suites has reduced the overall software development time and cost.

5. ACKNOWLEDGMENTS

Authors are grateful to the Vice-Chancellor (Prof. P.K. Bharti), Maharishi University of Information Technology Lucknow for providing the excellent facility in the computing lab (Maharishi university of Information Technology, Lucknow, India) for the entire research work. Thanks are also due to University Grant Commission, India for support to the University.

6. REFERENCES

[1] Bhasin H., Gupta S. and Kathuria M., "Regression Testing Using Fuzzy Logic", International Journal of Computer Science and Information Technologies 2013; Vol. 4 Issue 2, pp. 378-380.

[2] Mudgal P.A., "A Proposed Model for Minimization of test Suite", Journal of Nature Inspired Computing, 2013, Vol. 1, Issue 2, pp. 34-37.

[3] Alakeel M.A., "A Fuzzy Test Case Prioritization Technique for Regression Testing Programs with Assertions", The Sixth International Conference on Advanced Engineering Computing and Application in Sciences 2012; 78-82.

[4] Haider AA, Nadeem A, Akram S, "Regression Test Suite Optimization Using Adaptive Neuro Fuzzy Inference System", Frontiers of Information Technology 2016, pp. 52-56.

[5] Haider AA, Rafiq S, Nadeem A, "Test Suite Optimization Using Fuzzy Logic", International Conference on Emerging Technologies, 2012, pp. 1-6.

[6] Shamshiri A., Rojas M.J., Fraser G. and McMinn P., "Random or Genetic Algorithm search for Object-Oriented Test Suite Generation", Conference on Genetic and Evolutionary Computation 2015; 1367-1374.

[7] Qiu D, Li B, Ji S, Lenung H, "Rgeression Testing of Web Service: A Systematic Mapping Study", ACM Computing Surveys 2015, Vol. 47, Issue 2.

[8] Kumar G., and Bhatia K.P., "Software Testing Optimization Through Test Suite Reduction Using Fuzzy Clustering", Springer CSIT. 2013, Vol. 1, Issue 3, pp. 253-260.

[9] Usaola P.M., Mateo R.P. and Lamancha P.B., "Reduction of Test Suites Using Mutation", Springer-Verlag Berlin Heidelberg 2012, Vol. 72, Issue 12, pp. 425-438.

[10] Singh S, Shree R, "A Combined Approach to Optimize the Test Suite Size in Regression Testing", CSI Transaction on ICT, 2016, Vol. 4 Issue 2, pp. 73-78

[11] Asoudeh N. and Labiche Y., "A multi-Objective Genetic Algorithm for Generating Test Suites from Extended Finite State Machine", International Symposium on Search Based Software Engineering 2013; pp. 288-293.

[12] Ma YX, Sheng KB, Ye GC, "Test-Suite Reduction Using Genetic Algorithm", International Workshop on Advanced Parallel Processing Technologies 2005, pp. 253-262.

[13] Selva kumar S., Dinesh C.R.M. Dhinesh kumar C, and Ramraj N, "Reducing the Size of the Test Suite by Genetic Algorithm and Concept Analysis", Recent Trends in Network and Communications; 153-161.

[14] Dinca L, "Multi-Objective Test Suite Optimization for Event-B Models", International Conference on Informatics Engineering and Information Science 2011, pp. 551-565.

[15] Raamesh L, Uma VG, "Data Mining Based Optimization of test Cases to Enhance the Reliability of the Testing", Advances in Computing and Information Technology; pp. 89-98.

[16] Patton R, "Software Testing", Pearson Education 2001.

[17] Mondal K.S. and Tahbildar H., "Regression Test Cases Minimization for Object Oriented Programming using New Optimal Page Replacement Algorithm", International Journal of Software Engineering and Its Applications 2014, Vol. 8, Issue 6, pp. 253-264.

[18] Subhashini B, JeyaMala D, "Reduction of Test Cases Using Clustering Technique", International Journal of Innovative Research in Science, Engineering and Technology 2014, Vol. 3, Issue 3, pp. 1992-1996.

[19]